QNX 4 Operating System

Utilities Reference

For QNX 4.25 or later

© 2005, QNX Software Systems

© 1996 – 2005, QNX Software Systems. All rights reserved.

Published under license by:

QNX Software Systems Co.
175 Terence Matthews Crescent
Kanata, Ontario
K2M 1W8
Canada
Voice: +1 613 591-0931
Fax: +1 613 591-3579
Email: **info@qnx.com**
Web: **http://www.qnx.com/**

Publishing history	
July 1996	Second edition
January 1998	Third edition
August 1998	Fourth edition

Electronic edition published 2005.

Technical support options

To obtain technical support for any QNX product, visit the **Technical Support** section in the **Services** area on our website (**www.qnx.com**). You'll find a wide range of support options, including our free web-based **Developer Support Center**.

QNX, Momentics, Neutrino, and Photon microGUI are registered trademarks of QNX Software Systems in certain jurisdictions. All other trademarks and trade names belong to their respective owners.

Acknowledgments and Copyrights

The following individuals and organizations have made the utilities listed below available. We have enhanced some of these utilities and hope you will find them helpful. The full source code for these utilities, along with additional notes and credits, is available from our online update and conferencing system.

vedit is a product of Greenview Data, Inc.

awk, **ci**, **co**, **dd**, **diff3**, **gmake**, **gunzip**, **gzip**, **ident**, **pr**, **rcs**, **rcsclean**, **rcsdiff**, **rcsmerge**, **rlog**, **zcat** — The Free Software Foundation (see above); you can redistribute these and/or modify them under the terms of the GNU General Public License (GPL) as published by the Free Software Foundation. You can get the full source code and the GNU GPL in electronic form from the Free Software Foundation, 675 Mass Ave., Cambridge, MA 02139, USA.

elvis, **elvrec**, **ctags** — Steve Kirkendall

freeze, **melt**, **fcat** — Leonid A. Broukhis

less, **more** — Mark Nudelman

flex (**lex**), **yacc** — Vern Paxton, University of California, Berkeley

sed — Diomidis Spinellis. University of California, Berkeley

cpio, **pax**, **tar** — Mark H. Colburn, sponsored by the USENIX Association

pdksh (**sh**) — Eric Gisin

Contents

About This Reference xiii

What's new (as of May 15, 2003) xv

Utility Conventions 1

Syntax conventions 3

Interpreting utility syntax 3

Invoking utilities 4

File conventions 6

Signal conventions 6

Exit status conventions 6

Error conventions 7

Utilities Summary 9

Servers and Resident Processes 11

Drivers 13

Utilities 16

QNX 4 Utilities Reference — A to M 29

alive 33

Audio 36

awk 39

basename 63

bc 65

Blkfsys 76

boot 78

buildqnx 83

cat 93

cbpe 95
cd 98
cfont 102
chgrp 105
chkfsys 107
chmod 113
chown 117
ci 120
cksum 129
clock 132
cmp 134
co 136
comm 148
cp 150
cpio 165
cron 169
crontab 172
ctags 177
cut 179
date 182
dcheck 189
dd 194
Debugger32 197
Dev 204
Dev.ansi 207
Dev.con 235
Dev.par 243
Dev.pty 247
Dev.ser 251
df 257
diff 262
diff3 268

dinit 280
dirname 284
disktrap 286
ditto 289
Dosfsys 297
du 302
dumper 305
echo 308
efsinit 311
Efsys.365sl 313
Efsys.file 315
Efsys.pcmcia 317
Efsys.ram 319
egrep 322
elvis 323
elvrec 359
emu87 361
env 363
errno 365
esh 367
expand 374
expr 377
false 381
Fatfsys 382
fcat 389
fdformat 390
fdisk 395
fgrep 402
file 403
find 408
finger 431
fold 434

freeze 436
Fsys 440
Fsys.aha2scsi 447
Fsys.aha4scsi 456
Fsys.aha7scsi 467
Fsys.aha8scsi 477
Fsys.amdscsi 486
Fsys.ata 494
Fsys.eide 498
Fsys.floppy 505
Fsys.ide 509
Fsys.mitsumi 513
Fsys.ncr8scsi 515
Fsys.ps2scsi 516
Fsys.sym8scsi 524
Fsys.us14scsi 534
Fsys.vpm50 542
fsysinfo 545
fullpath 550
gmake 552
grep 553
gunzip 561
gzip/**gunzip**/**zcat** 562
hd 567
head 571
id 573
ident 576
infocmp 578
Input 580
inputtrap 589
/etc/install 593
int10 595

Iso9660fsys 598
join 604
kbd 607
kedit 609
kill 611
kill_vcs 614
ksh 616
less 617
lex 633
license 639
licinfo 643
ln 645
lockfd 648
logger 650
login 652
logname 655
logout 656
lp 657
lpc 660
lpq 663
lprm 666
lpsrvr 668
ls 670
/etc/magic 677
mailx 681
make 701
melt 721
mkcis 722
mkdir 728
mkffs 731
mkfifo 734
mkipl 736

modem 739
monitor 748
more 749
mount 753
mount_cdfs 757
mount_dos 758
mount_iso9660 760
mount_smb 762
Mouse 766
mousetrap 769
Mqueue 770
msgprint 772
mv 773

QNX 4 Utilities Reference — N to Z 777

nameloc 781
Net 784
Net.arc20 789
Net.arcnet 792
Net.arcxir 795
Net.crys8900 799
Net.epic 802
Net.ether1000 806
Net.ether2100 811
Net.ether21x4x 817
Net.ether503 818
Net.ether509 823
Net.ether589 827
Net.ether7231 831
Net.ether8003 835
Net.ether82557 839
Net.ether82595 843
Net.ether8900 847

Net.ether9000 848
Net.ether905 852
Net.fd 857
Net.fddidfe 861
Net.i82540 864
Net.ns83815 867
Net.rtl 871
Net.sis9 876
Net.soho 880
Net.solotr 883
Net.sym 886
Net.tr164a 890
Net.tulip 893
Net.via 900
netboot 905
netinfo 914
netmap 916
netpoll 922
netsniff 925
nettrap 928
newgrp 931
nice 933
noditto 936
nohup 938
on 940
passwd 943
paste 948
pax 951
pc-pin 959
pccard 961
pccard-launch 964
PcmciaBeep 967

PcmciaLaunch 969
Pcmcia.* 972
pcmciatrap 987
pin 989
Pipe 993
pr 995
prefix 1000
printf 1004
Proc32 1011
ps 1018
pwd 1023
qcp 1024
qed 1028
qtalk 1030
ramtrap 1047
rcs 1049
rcsclean 1055
rcsdiff 1058
rcsmerge 1061
renice 1064
rlog 1066
rm 1072
rmdir 1075
romqnx 1077
rtc 1080
sac 1084
scsi_util 1087
sed 1091
sh 1101
show_pci 1159
shutdown 1165
sin 1167

sinit 1191
size 1194
slay 1196
sleep 1200
Slib 1202
SMBfsys 1204
sort 1210
spatch 1214
split 1218
strings 1221
stty 1223
su 1234
sync 1235
syslogd 1237
tail 1242
tape 1245
tar 1248
tee 1251
termdef 1253
textto 1258
tic 1260
ticksize 1262
time 1264
tinit 1266
touch 1270
tr 1273
tracectrl 1276
traceinfo 1279
tracelogger 1300
true 1303
tty 1304
uesh 1305

umask 1311
umount 1315
uname 1317
unexpand 1319
uniq 1321
use 1324
usemsg 1327
user_smb 1331
vdir 1333
vedit 1335
vi 1350
vol 1351
wall 1354
wc 1357
which 1359
who 1361
write 1363
xargs 1366
yacc 1369
zap 1373
zcat 1376

Index 1377

About This Reference

The *QNX 4 Utilities Reference* describes the utilities and server processes included with the QNX 4 operating system. This reference is intended for all users — from end-users to system administrators.

The current printed edition is a two-volume set:

1. The first volume (*Utilities Reference — A to M*) contains:
   - a section on the conventions used in this reference
   - a summary table of all the utilities
   - the documentation for the utilities and OS components from **alive** to **mv**.
2. The second volume (*Utilities Reference — N to Z*) covers the utilities **nameloc** to **zcat**.

For an overview of the changes to the software and docs, see What's new (as of May 15, 2003).

This reference includes:

- a section on the conventions used here
- a summary table of all the utilities
- the documentation for the utilities and OS components from **alive** to **zcat**.

What's new (as of May 15, 2003)

New:

Net.i82540	Driver for Intel 82540 Gigabit Ethernet cards.
Net.sis9	Driver for SiS 900 Ethernet cards.
Net.soho	Driver for Macronix MX98715 Ethernet network cards.
pccard	Server for PCMCIA/CardBus (PC Card).

pccard-launch

Automatically starts drivers when PC Cards are inserted.

pc-pin Displays PC Card information.

Other changes:

vedit The names of these configuration files no longer start with a period:

- **veditrc**
- **vedit.cfg**
- **vedit.kbd**
- **vedit.key**

For more information about the changes to vedit, see **/qnx4/vedit/whatsnew.txt**.

Utility Conventions

Syntax conventions

Most QNX utilities follow standard conventions for argument syntax and behavior. These conventions are based on the utility conventions outlined in POSIX 1003.2-1992.

The syntax synopsis for each utility appears at the top of the page of its manual entry. The utility name appears first, followed by other allowed command-line arguments, which include options, option arguments (e.g. "*number*" in **-n** *number*), and operands (e.g. the names of files to act on).

The syntax synopsis is the only reliable source for information about mutual exclusivity of options and about whether a command-line element is optional or required. This information isn't usually contained in the detailed option listings that appear after the syntax section.

A typical utility syntax line looks like this:

utilityname [-abcd] [-o *arg* **| -p** *arg***]** *infile***...** *outfile*

The example above shows a utility called **utilityname** that accepts the options **-a**, **-b**, **-c**, and **-d** — these options may be used alone or in any combination.

The utility also accepts the options **-o** and **-p**, both of which require an option argument, and which may not be used together (but may be used with the other options **-abcd**). The utility requires two or more operands: one or more *infile* and exactly one *outfile*.

Interpreting utility syntax

Here are the main principles at work:

- When utilities have many options, the options may appear grouped together in the syntax like this:
  utilname [-abcd]

  which means that the options **-a**, **-b**, **-c**, and **-d** are supported.

- Options, option arguments, and operands enclosed in brackets (**[** and **]**) are optional and can be omitted. Note that the **[** and **]** symbols should never be included in the actual command.
- Arguments separated by **|** are mutually exclusive. Sometimes mutually exclusive arguments that relate to modes of operation will be indicated with multiple syntax lines representing the different forms of the command.
- A trailing ellipsis mark (**...**) after options or operands indicates that the preceding item may be repeated. If the preceding item is optional, the ellipsis indicates that the item may occur zero or more times, e.g.:

  utility [*filename***...]**

  If the item is mandatory, the ellipsis indicates it may occur one or more times, e.g.:

  utility *filename***...**

Invoking utilities

There are a number of general guidelines to follow when running utilities:

- An option may be followed by another option after a single dash (**-**) on the command line as long as each preceding option doesn't have an option argument. For example, the option string **-abc** is equivalent to **-a -b -c**. However, if **-a** accepts an option argument, then **-abc** would be equivalent to **-a bc** instead.
- Options and their option arguments should be specified with spacing as shown in their documentation. If the documentation says:

  -n *number*

  the *number* should be a separate command-line argument from the **-n**. But if the documentation refers to:

  -n*number*

then *number* should appear in the same argument as **-n** without any intervening blanks. Utilities in QNX and in POSIX-conforming systems will permit both forms in all utilities *unless otherwise stated*, but greatest portability is achieved by using the preferred form. This is particularly important when developing scripts that may be used on multiple (QNX and non-QNX) platforms.

- Options are usually listed in alphabetical order, but there's no restriction on the order that they may appear in the command line when used, unless otherwise indicated in the documentation for the utility. Note that in some utilities, mutually exclusive options will override each other in a "last one wins" manner.

- All options and associated option arguments must precede any operands on the command line. For example, if you want to run the **cp** utility with the **-R** option, you may enter:

  ```
  cp -R dir1 dir2
  ```

  but not:

  ```
  cp dir1 dir2 -R
  ```

- Decimal integers will be accepted when numeric values are required in operands and option arguments, unless otherwise specified. Some utilities may support **0***octal* and **0x***hex* numbers as well without being documented as doing so. For this reason, don't precede decimal numbers with leading zeros.

- Integer numerical operands and option arguments must be in the range 0 to 2147483647 unless otherwise specified. If negative numbers are accepted, the acceptable range is -2147483647 to 2147483647.

- The argument **--** ("dash dash") may be placed on the command line as a delimiter indicating the end of options and the start of operands. This is particularly useful when the operands themselves might start with a dash. For example, to remove a file named "**-t**", you would use:

```
rm -- -t
```

Utilities that don't accept any options will also accept and discard a `--` before their operands, unless otherwise indicated.

- Most utilities that accept filenames as operands (and sometimes as option arguments) will accept the filename "`-`" to mean standard input, or, when unambiguous from its context, standard output.

File conventions

File pathnames specified on the command line are restricted to 255 characters. Some input files are specifically identified as "text files." Text files are expected to contain ASCII text in newline-terminated lines that don't exceed 2048 characters, unless otherwise indicated.

Signal conventions

Signal actions are inherited from the process that invokes the utility. Most utilities don't do any special processing upon receipt of a signal, but behave instead according to the system defaults. When a utility performs some action on receipt of a signal other than the default, it will be documented as doing so.

Note that temporary files will *not* be left in place after a utility is terminated due to a signal, unless otherwise specified.

Utilities that begin with a capital letter are used to denote servers and resident processes. All other utility names are entirely in lowercase. Servers and resident processes will typically run only as root, and will ignore most signals (such as SIGPWR).

Exit status conventions

Utilities normally return zero for successful completion and values greater than zero when unsuccessful. Some utilities will return different non-zero numbers according to the reason they failed. Beware of testing for a specific non-zero number to indicate failure. (In most cases utilities that may return different non-zero numbers are

explicitly documented as doing so. However, you should not rely on this.)

For some utilities, the exit status may reflect only the success or failure of the last action taken (of many). In these cases, this behavior will be explicitly documented in the “Exit status” section.

Error conventions

Utilities may fail for many reasons ranging from incorrect usage to underlying system failure. The documentation for the utilities doesn’t attempt to outline the exact behavior for all possible modes of failure.

In all cases, unless otherwise specified, every error will result in a diagnostic message printed to standard error.

When an error occurs, the utility will stop the processing of the current operand and proceed to process the next operand in the sequence. If a utility fails to process one operand but succeeds on others, the exit status will still reflect failure. For utilities that recurse through a filesystem (e.g. **`find`**), if an action cannot be performed on one file within a hierarchy, the utility will stop processing that file and go on to the subsequent files in the hierarchy.

When an unrecoverable error occurs (e.g. insufficient memory), the utility will print a diagnostic message to standard error and exit immediately.

Utilities Summary

Servers and Resident Processes

Name	Type	Description
Audio	manager	Audio device manager
Blkfsys	manager	Block special filesystem
boot	bootstrap	Bootstrap the operating system
cron	manager	Clock server
Debugger32	debugger	Low-level system debugger
Dev	manager	Device manager
Dev.*	driver	Device drivers (see “Drivers”)
disktrap	utility	Autodetect disk controller and start appropriate driver
Dosfsys	manager	DOS filesystem manager
Efsys.365sl	manager	Embedded filesystem manager for Intel 82365 PCMCIA adapter
Efsys.file	manager	Embedded filesystem in a file
Efsys.pcmcia	manager	Generic embedded filesystem driver options
Efsys.ram	manager	Embedded filesystem manager for generic RAM area
emu87	manager	Start 80x87 floating-point emulator
Fatfsys	manager	DOS FAT12/16/32 filesystem manager
Fsys	manager	Filesystem manager
Fsys.*	driver	Filesystems drivers (see “Drivers”)
Input	manager	Input manager for GUIs

continued…

Name	Type	Description
inputtrap	utility	Detect input devices and start Input Manager
int10	manager	Provide some of the BIOS INT10 functions
Iso9660fsys	manager	ISO 9660 filesystem manager
lpsrvr	manager	Print server
modem	utility	Modem handler to answer calls
Mouse	manager	Mouse driver
mousetrap	utility	Autodetect mouse and optionally start correct driver
Mqueue	manager	POSIX message queue server
nameloc	manager	Name locator
Net	manager	Network manager
Net.*	driver	Network drivers (see “Drivers”)
nettrap	utility	Autodetect network card and start appropriate driver
netboot	manager	Service boot requests from the network
pccard	manager	PCMCIA/CardBus (PC Card) server
pccard-launch	manager	Automatically start drivers when PC Cards are inserted
Pcmcia.*	manager	PCMCIA servers for various chipsets
PcmciaBeep	manager	Beep PC speaker when PCMCIA cards are changed

continued...

Name	Type	Description
PcmciaLaunch	manager	Automatically start drivers when PCMCIA cards are plugged in
pcmciatrap	utility	Autodetect PCMCIA adapter and optionally start correct driver
Pipe	manager	Pipe manager
Proc32	manager	Process manager
ramtrap	utility	Automatically add memory beyond what the BIOS reports
sinit	utility	System initialization
Slib*	library	System shared libraries
SMBfsys	manager	SMB filesystem client
syslogd	manager	System message logger
tinit	manager	Terminal initialization
tracelogger	utility	Extract trace events and log them to a file

Drivers

Device Drivers

Name	Type	Description
Dev.ansi	driver	ANSI console
Dev.con	driver	QNX console
Dev.par	driver	Parallel devices
Dev.pty	driver	Pseudo ttys
Dev.ser	driver	Serial devices

Filesystem Drivers

Name	Type	Description
Fsys.aha2scsi	driver	Adaptec AIC-6260/6360-based SCSI host adapters
Fsys.aha4scsi	driver	Adaptec 154x and compatible SCSI host adapters
Fsys.aha7scsi	driver	Adaptec AIC-7770/7870/7880-based SCSI host adapters
Fsys.aha8scsi	driver	Adaptec common HIM SCSI adapters
Fsys.amdscsi	driver	AMD 53c974/79c974 PCI SCSI host adapter
Fsys.ata	driver	ATA fixed-disk interface
Fsys.eide	driver	ATA/IDE disk interface and ATAPI CD-ROM
Fsys.floppy	driver	Floppy disk
Fsys.ide	driver	IDE fixed-disk interface
Fsys.mitsumi	driver	Mitsumi LU-005 and FX-001 CD-ROM drives
Fsys.ncr8scsi	driver	NCR 53c810/815/820/825 PCI SCSI host adapter
Fsys.ps2scsi	driver	PS/2 SCSI host adapter
Fsys.sym8scsi	driver	Symbios/NCR 53c8xx SCSI host adapter
Fsys.us14scsi	driver	Ultrastor US-14F/34F SCSI host adapter

Network Drivers

Name	Type	Description
Net.arc20	driver	SMC COM20020 Arcnet network driver
Net.arcnet	driver	Arcnet network driver
Net.arcxir	driver	Xircom Pocket Arcnet Adapter II network driver
Net.crys8900	driver	Crystal 8900/8920 Ethernet network driver
Net.epic	driver	SMC 83C170 Ethernet network driver
Net.ether1000	driver	NE1000/2000 Ethernet network driver
Net.ether2100	driver	NE1500T/2100 Ethernet network driver
Net.ether21x4x	driver	See **Net.tulip**.
Net.ether503	driver	3Com 503 Ethernet network driver
Net.ether509	driver	3Com 509 Ethernet network driver
Net.ether589	driver	3Com 3C589 PCMCIA Ethernet network driver
Net.ether7231	driver	AT&T 7231 Ethernet network driver
Net.ether8003	driver	WD/SMC 8003 Ethernet network driver
Net.ether82557	driver	Intel 82557/8 Ethernet network driver
Net.ether82595	driver	Intel 82595 Ethernet network driver
Net.ether8900	driver	See **Net.crys8900**.

continued...

Name	Type	Description
Net.ether9000	driver	SMC91C90/91C92 Ethernet network driver
Net.ether905	driver	3Com 905 Ethernet network driver
Net.fd	driver	File-descriptor-based network driver
Net.fddidfe	driver	DEC FDDI network driver
Net.i82540	driver	Intel 82540 Gigabit Ethernet network driver
Net.rtl	driver	Realtek Semiconductor RTL8139A Ethernet network driver
Net.sis9	driver	SiS 900 Ethernet network driver
Net.soho	driver	Macronix MX98715 Ethernet network driver
Net.solotr	driver	IBM PCI Token-Ring network driver
Net.tr164a	driver	IBM Token Ring 16/4 Adapter network driver
Net.tulip	driver	DEC 21x4x Ethernet Network Driver
Net.via	driver	VIA Technologies VT86C100A Ethernet network driver

Utilities

Utility	Class	Description
alive	QNX	Output list of nodes
awk	POSIX	Pattern scanning and processing language

continued...

Utility	Class	Description
basename	POSIX	Return nondirectory portion of pathname
bc	POSIX	Arbitrary-precision arithmetic language
buildqnx	QNX	Build a custom version of QNX
cat	POSIX	Concatenate and print files
cbpe	QNX	Use Byte-Pair Encoding to compress or decompress a file
cd	POSIX	Change working directory
cfont	QNX	Load text-mode font data into EGA or VGA cards
chgrp	POSIX	Change file group ownership
chkfsys	QNX	Check an entire QNX filesystem for consistency
chmod	POSIX	Change file modes
chown	POSIX	Change file ownership
ci	UNIX	Check in RCS revisions
cksum	POSIX	Display file checksums and block counts
clock	QNX	Display date and time on console
cmp	POSIX	Compare two files
co	UNIX	Check out RCS revisions
comm	POSIX	Select or reject lines common to two files
cp	POSIX	Copy files

continued...

Utility	Class	Description
cpio	UNIX	Copy file archives in and out
cron	UNIX	Clock server
crontab	POSIX	Schedule periodic background work
ctags	POSIX	Generate tags files
cut	POSIX	Cut out selected fields of each line of a file
date	POSIX	Display or set the date and time
dcheck	QNX	Check a disk for bad blocks
dd	UNIX	Convert a file while copying it
df	POSIX	Report disk space free
diff	POSIX	Report the difference between two files
diff3	GNU	Show differences among three files
dinit	QNX	Initialize a QNX filesystem
dirname	POSIX	Return directory portion of pathname
disktrap	QNX	Autodetect disk controller and start appropriate driver
ditto	QNX	View or control another console
du	POSIX	Estimate disk space usage
dumper	QNX	Dump postmortem state of a program
echo	POSIX	Write arguments to standard output

continued...

Utility	Class	Description
efsinit	QNX	Erase Flash partitions or format an embedded filesystem
egrep	UNIX	Extended regular expression **grep**
elvis	UNIX	Visual display editor clone
elvrec	UNIX	Recover **elvis** documents after a crash
env	POSIX	Set environment for command invocation
errno	QNX	Explain errno numbers
esh	QNX	Embedded shell
expand	POSIX	Convert tabs to spaces
expr	POSIX	Evaluate arguments as an expression
false	POSIX	Return false value
fcat	UNIX	Display compressed (frozen) files
fdformat	QNX	Format floppy diskettes
fdisk	QNX	Create QNX disk partition
fgrep	UNIX	Fixed string **grep**
file	UNIX	Determine file type
find	POSIX	Find files
finger	UNIX	Display information about a user
fold	POSIX	Fold lines
freeze	UNIX	Compress and uncompress files

continued...

Utility	Class	Description
fsysinfo	QNX	Display filesystem statistics
fullpath	QNX	Display network-qualified pathnames
gmake	GNU	Maintain, update, and regenerate groups of programs
grep	POSIX	Search for string patterns
gunzip	GNU	Uncompress files; see **gzip**
gzip/gunzip/zcat	GNU	Compress or expand files
hd	UNIX	Display files in decimal, hex, octal, or ASCII
head	POSIX	Copy the first part of files
id	POSIX	Return user ID
ident	UNIX	Identify files
infocmp	UNIX	Output the contents of a terminfo capability file
inputtrap	QNX	detect input devices and start Input Manager
/etc/install	QNX	Install a software package
join	POSIX	Relational database operator
kbd	QNX	Specify keyboard layout
kedit	QNX	Edit a keyboard layout file
kill	POSIX	Terminate or signal processes
kill_vcs	QNX	Kill all virtual circuits to a node
ksh	UNIX	Korn Shell command interpreter
less	UNIX	Interactive paginator

continued...

Utility	Class	Description
lex	POSIX	Lexical analyzer generator
license	QNX	Install/transfer software licenses
licinfo	QNX	List licenses in use
ln	POSIX	Create links to (aliases for) files
lockfd	QNX	Lock a floppy drive to one capacity/media type
logger	POSIX	Make entries in the system log
login	UNIX	Log in
logname	POSIX	Return user's login name
logout	UNIX	Terminate a session
lp	POSIX	Send files to a printer
lpc	UNIX	Line printer control program
lpq	UNIX	Display print queue status
lprm	UNIX	Remove jobs from the printer queue
ls	POSIX	List directory contents
mailx	POSIX	Read or send mail messages
make	POSIX	Maintain, update, and regenerate groups of programs
melt	UNIX	Uncompress frozen files
mkcis	QNX	Create a Card Information Structure
mkdir	POSIX	Make directories
mkffs	POSIX	Make a Flash (embedded) filesystem image

continued...

Utility	Class	Description
mkfifo	POSIX	Make FIFO special files
mkipl	QNX	Convert a binary image into ROM format
monitor	QNX	Generate kernel event log
more	POSIX	Display files on a page-by-page basis
mount	UNIX	Mount partitions and filesystems
mount_cdfs	QNX	Mount CD-ROM filesystems
mount_dos	QNX	Mount DOS filesystems
mount_iso9660	QNX	Mount ISO 9660 filesystems
mount_smb	QNX	Mount SMB filesystems
mousetrap	QNX	Autodetect mouse and start appropriate driver
msgprint	QNX	Display kernel event log
mv	POSIX	Move files
netinfo	QNX	Display network events
netmap	QNX	Load the Network Manager's logical-to-physical node ID mapping table
netpoll	QNX	Display or change default polling parameters of a node
nettrap	QNX	Autodetect network card and start appropriate drivers
newgrp	POSIX	Change to a new group
nice	POSIX	Run a program at an altered priority

continued...

Utility	Class	Description
noditto	QNX	Prevent another user from dittoing your machine
nohup	POSIX	Invoke a command immune to hangups
on	QNX	Execute a command on another node or tty
passwd	UNIX	Change login password
paste	POSIX	Merge corresponding or subsequent lines of files
pax	POSIX	Portable archive interchange
pc-pin	QNX	Display PC Card information
pcmciatrap	QNX	Autodetect PCMCIA adapter and optionally start correct driver
pin	QNX	Display PCMCIA information
pr	POSIX	Print files
prefix	QNX	Examine or change the pathname prefix tree
printf	POSIX	Write formatted output
ps	POSIX	Report process status
pwd	POSIX	Print working directory name
qcp	QNX	QNX communications protocol
qed	QNX2	QNX fullscreen editor
qtalk	QNX	Talk over communications line
ramtrap	QNX	Automatically add memory beyond what the BIOS reports

continued...

Utility	Class	Description
rcs	UNIX	Change RCS file attributes
rcsclean	UNIX	Clean up working files
rcsdiff	UNIX	Compare RCS revisions
rcsmerge	UNIX	Merge RCS revisions
renice	POSIX	Adjust the priority of a running process
rlog	UNIX	Print log messages and other info about RCS files
rm	POSIX	Remove files
rmdir	POSIX	Remove directories
romqnx	QNX	Convert a **buildqnx** image to a ROM image
rtc	QNX	Set or get date from realtime clock
sac	QNX	Display system activity at each priority level
scsi_util	QNX	Control SCSI and EIDE devices
sed	POSIX	Stream editor
sh	POSIX	Command interpreter
show_pci	QNX	Display PCI devices
shutdown	QNX	Shut down and reboot the system
sin	QNX	Display system information
size	UNIX	Print section sizes of object and load modules

continued...

Utility	Class	Description
slay	QNX	Kill or modify a process by name
sleep	POSIX	Suspend execution for an interval
sort	POSIX	Sort, merge, or sequence-check text files
spatch	QNX	Fullscreen patch utility
split	POSIX	Split files into pieces
strings	POSIX	Find printable strings in files
stty	POSIX	Set tty attributes
su	UNIX	Switch user ID
sync	UNIX	Update filesystems to match cached data
tail	POSIX	Copy the last part of a file
tape	QNX	Control a tape driver
tar	UNIX	Read and write tape archive files
tee	POSIX	Duplicate standard input
termdef	QNX	Display or set the terminal type
textto	QNX	Convert text files to QNX 2, QNX 4, or DOS format
tic	UNIX	Terminfo compiler
ticksize	QNX	Set the system timer granularity
time	UNIX	Determine the execution time of a command
tinit	QNX	Terminal initialization

continued...

Utility	Class	Description
touch	POSIX	Change file access and modification times
tr	POSIX	Translate characters
tracectrl	QNX	Display and change trace parameters
traceinfo	QNX	Print trace events
tracelogger	QNX	Extract trace events and log them to a file
true	POSIX	Return true value
tty	POSIX	Return user's terminal name
uesh	QNX	Micro-embedded shell
umask	POSIX	Get or set the file mode creation mask
umount	UNIX	Unmount filesystems or partitions
uname	POSIX	Return system name
unexpand	POSIX	Convert spaces to tabs
uniq	POSIX	Report or filter out repeated lines in a file
use	QNX	Print a usage message
usemsg	QNX	Change usage message for a command
user_smb	QNX	Map a QNX user to an SMB user
vdir	QNX	Create virtual directories
vedit	QNX	Full screen editor

continued...

Utility	Class	Description
vi	UNIX	Visual display editor
vol	QNX	Manage multi-volume archives
wall	UNIX	Broadcast a message
wc	POSIX	Word, line, and byte count
which	UNIX	Locate a program file
who	POSIX	Display who is on the system
write	POSIX	Write a message to a user's terminal
xargs	POSIX	Construct argument list(s) and execute program
yacc	POSIX	A parser generator
zap	QNX	Destroy a damaged file
zcat	GNU	Concatenate compressed files; see **gzip**

QNX 4 Utilities Reference — A to M

© 2005, QNX Software Systems

This volume (*Utilities Reference — A to M*) contains documentation for the utilities and OS components from **alive** to **mv**.

The second volume (*Utilities Reference — N to Z*) covers the utilities **nameloc** to **zcat**.

Syntax:

alive [-aehv] [-c *columns***]**

alive -u

Options:

-a	Show all nodes (**alive** normally stops after the last non-masked node).
-c *columns*	Display nodes in *columns* columns (default is 6).
-e	Don't show masked nodes in the list (overrides the **-a** option).
-f	Force an attempt to talk to all nodes, even those currently marked as down in *qnx_net_alive()* list.
-h	Horizontal (i.e. in rows) format
-u	Show only "Up" nodes, and display node numbers only. Nodes are displayed in a single column.
-v	Vertical (i.e. in columns) format. (default)

Description:

The **alive** utility outputs a list of nodes, and the status for each, to standard output. By default, the list is in table form (nodes are listed in columns). A node can be in one of the following states:

If a node is:	It means that the node:
Up	is up and that **alive** was able to create a virtual circuit (VC) to its **Proc** manager.

continued...

If a node is:	It means that the node:
Down	isn't up, isn't running QNX 4, or is marked as "down" in the *qnx_net_alive()* list. See Caveat.
Mask	has been masked using the **netmap** utility. Normally, this is an Arcnet node that runs DOS or QNX, for which an attempt to create a virtual circuit would result in a timeout.

Examples:

Show the status of all nodes, up to the last non-masked node:

```
alive
```

For every node alive on the network, put a list of its serial ports into the file **allser**:

```
alive -u | xargs -in1 ls -l //{}/dev/ser\* >allser
```

Files:

The **alive** utility writes the alive list to standard output. Any errors that occur will result in diagnostic messages to standard error. Standard input is not used.

Exit status:

0 Successful completion.

>0 An error occurred.

Caveats:

The **alive** utility will communicate only with the nodes present in the Network manager's node-mapping table, and which are listed in the *qnx_net_alive()* list which is maintained by the name locators. The

name locator can be fooled if the name locators have not been correctly distributed on the network (see **nameloc**). In this case, the **-f** option can be specified to force **alive** to make an attempt to talk to every node.

See also:

nameloc, **Net**, **netinfo**, **netpoll**, **Proc**

Audio

© 2005, QNX Software Systems

Audio device manager (QNX)

Syntax:

Audio [*options***] [***driver* **[***driver_options***]]...**

Options:

-A Use auto-initialize DMA.

-D Debug mode.

-F Run with priority inheritance

driver [*driver_options*]...

Specify the driver to use and the driver options to match the hardware configuration being used. Available drivers include:

driver	Hardware	Default options
sb	Soundblaster	**-b220 -d1 -i7**
pas	ProAudio Spectrum	**-b388 -d1 -i7**
mss	Microsoft Sound System	**-b530 -d1 -i7**
mpu	Roland MPU-401	**-b330 -i2**
adlib	Adlib	**-b388**

Consult the usage message (**use Audio**) or the hardware support technical notes on QUICS for recent additions to the list of supported audio hardware types.

driver_options may be specified to override the defaults listed in the table above. Available options are:

-b *port* Specify base I/O port.

-d *dma1*[**,***dma2*]

Specify DMA channel(s), if applicable.

-i *irq* Specify IRQ, if applicable.

Description:

The **Audio** manager handles playing and recording sounds on common audio hardware through the device **/dev/dsp**. Recording is done by reading from **/dev/dsp** and playback is done by writing to **/dev/dsp**. Recording and playback parameters may be adjusted via C programs through *ioctl()* calls. Some of the most commonly adjusted parameters include volume level, sample rates and sample resolution.

A technical note on audio programming can be found in the **/etc/readme/technotes/** directory which documents the available *ioctl()* calls, and demo source code is shipped with the operating system under the **/usr/demo/src/audio** directory.

Examples:

Start **Audio** using a Soundblaster card and default options:

```
Audio sb &
```

Start **Audio** using a Soundblaster card in the default configuration. Record a test audio sample into **mysounds** and play it back:

```
Audio sb &
cat /dev/dsp >mysounds
```

Stop the sampling operation by typing Ctrl – C.

Now play it back:

```
cat mysounds >/dev/dsp
```

Files:

Audio closes its standard input, standard output and standard error after initialization has been completed. Error messages during initialization will be written to standard error.

© 2005, QNX Software Systems

Audio adopts the character special file **/dev/dsp** with which applications may perform I/O in order to playback and record sound data.

Exit status:

Audio normally will not terminate. However, it is possible for it to exit if it encounters errors on startup, or if it has not been started in the background (in which case it will fork itself and the foreground copy will exit with an exit status of 0).

0 Not run in foreground; successfully forked a background copy.

>0 Failed to start successfully. A message stating the reason for the failure will be written to standard error before **Audio** exits.

See also:

/etc/readme/technotes/Audio.doc,
/usr/demo/src/audio/vplay,
/usr/demo/src/audio/vplay.c,
/usr/include/sys/audio.h

Syntax:

awk [-F *ere***] [-v** *var*=*val***] [-W** *GNU_extension***...]**
[--] *program* **[**_argument_**]...**

awk [-F *ere***] -f** *progfile* **[-v** *var*=*val***] [-W** *GNU_extension***...]**
[--] [_argument_**]...**

Options:

-f *progfile* Specifies the name of the **awk** program, *progfile*, which contains **awk** statements. You can also specify the **awk** program as a single argument in the command line by using quotes.

-F *ere* Define the input field separator (FS) to be the extended regular expression *ere*, before any input is read.

-v *var*=*val* Assign the value *val* to variable *var* before execution of the program begins.

-W *GNU_extension*[**,** *GNU_extension*]...

Specify a GNU extension. Multiple **-W** options may be specified, or multiple extensions may be listed in a single **-W** option if they are separated by commas (or white space if the entire option argument has been quoted from the command line). The GNU extended options are:

compat Run in compatibility mode. In compatibility mode, GNU **awk** behaves identically to UNIX **awk**; none of the GNU-specific extensions are recognized.

copyleft
copyright Print the short version of the GNU copyright information message on the error output.

help
usage Print a relatively short summary of the available options on the error output.

lint Provide warnings about constructs that are dubious or non-portable to other **awk** implementations.

posix This turns on compatibility mode, with the following additional restrictions:

- **\x** escape sequences are not recognized.
- The synonym **func** for the keyword **function** is not recognized.
- The operators ****** and ****=** cannot be used in place of **^** and **^=**.

source=*program-text*

Use *program-text* as AWK program source code. This option allows the easy intermixing of library functions (used via the **-f** option) with source code entered on the command line. It is intended primarily for medium to large size AWK programs used in shell scripts. The **-W source=** form of this option uses the rest of the command-line argument for program-text; no other options to **-W** will be recognized in the same argument.

version Print version information for this particular copy of GNU **awk** on the error output. This is useful mainly for knowing if the current copy of GNU **awk** on your system is up to date with respect to whatever the Free Software Foundation is distributing.

`--` Indicates end of options, in case subsequent item begins with dash (`-`) and is not an option.

program The text of an **awk** program that contains **awk** statements, assuming no **-f***progfile* option has been specified. The **awk** program can either be in a file *progfile* or specified in the command line, taking into account the quoting rules of the shell.

argument You can intermix either of these two types of arguments:

- *file* The pathname of a file that contains the input to be read; the input is matched against the set of patterns in the program. If you don't specify any files containing input, or if you specify the dash character (`-`), the standard input is used.
- *assignment* Expressions in the form *name=value* can be passed to **awk** where each instance of *name* represents the name of an **awk** variable. Each such variable assignment occurs just prior to the processing of the following *file*, if any. The assignment is done before the *file* argument is executed, and after the **BEGIN** action — if any — of that file.

Description:

The **awk** utility executes programs written in the **awk** programming language; this language specializes in manipulating textual data. An **awk** program is a sequence of patterns and corresponding actions. When input matching a specified pattern is read, the action associated with that pattern is carried out. The **awk** shipped with QNX is a port of **gawk** (GNU **awk**). The source code and original GNU documentation is available from the Free Software Foundation.

© 2005, QNX Software Systems

The **awk** utility interprets each input line as a sequence of fields, where, by default, a field is a string of non-blank characters. You can change this default white space delimiter by using the built-in variable, *FS* (see Variables), or the **-F** *ere* option. The **awk** utility denotes the first field in a line $*1*, the second $*2*, and so forth. A $*0* refers to the entire line; setting any other field causes $*0* to be reevaluated.

Each input line matched by the patterns and each input line for the *getline* function (see the section on Functions) is limited to 1024 bytes.

Programs in **awk** are composed of statements of the form:

pattern **{** *action* **}**

You can omit either the pattern or the action (that includes the enclosing braces). In the following sections of this description, blank characters between operators and reserved words are ignored, unless otherwise specified. All blank characters are significant inside literal strings and after a function name. A missing pattern matches any line of input, and a missing action is equivalent to an action that writes the matched line of input to standard output.

An **awk** program follows this general procedure:

1. Starts by executing the actions associated with all **BEGIN** patterns (**BEGIN** patterns are described in the section Special patterns — **BEGIN** and **END**).
2. Processes each file argument in turn — or standard input if no files are specified — by cyclically reading and saving data from the file until a record separator is seen, evaluating each pattern in the script and executing that action for patterns that evaluate to true (the default record separator is the **newline** character).
3. Executes the actions associated with all **END** patterns.

Expressions:

Expressions describe computations used in patterns and actions. Expressions in **awk** are constructed from such operators as conditionals, logicals, arithmetics, assignments, subscripts, and fields. Expressions take on string or numeric values appropriate to the context. The following table, "**awk** Expressions in Decreasing Precedence," displays valid expressions in groups, starting from the highest priority.

In the following table, the abbreviation *expr* represents any expression. The abbreviation *lvalue* represents any entity that can be assigned to (i.e. on the left side of an assignment operator).

Syntax	Description
(*expr***)**	grouping
$*expr*	references field number *expr*
++*lvalue*	pre-increment *lvalue* by 1
--*lvalue*	pre-decrement *lvalue* by 1
*lvalue***++**	post-increment *lvalue* by 1
*lvalue***--**	post-decrement *lvalue* by 1
expr **^** *expr*	exponentiation
! *expr*	logical not
+ *expr*	unary plus
- *expr*	unary minus
expr ***** *expr*	multiplication
expr **/** *expr*	division
expr **%** *expr*	integer modulus
expr **+** *expr*	addition

continued...

© 2005, QNX Software Systems

Syntax	Description
expr **-** *expr*	subtraction
expr expr	string concatenation of two *expr*s
expr **<** *expr*	less than
expr **<=** *expr*	less than or equal to
expr **!=** *expr*	not equal to
expr **==** *expr*	equal to
expr **>** *expr*	greater than
expr **>=** *expr*	greater than or equal to
expr1 **~** *expr2*	1 if *expr1* matches the ERE described by *expr2*
expr1 **!~***expr2*	1 if *expr2* doesn't match the ERE described by *expr2*
expr **in** *array*	1 if *array*[*expr*] exists
(*index***)** **in** *array*	handles multi-dimensional arrays
expr **&&** *expr*	logical AND
expr **||** *expr*	logical OR
*expr***?***expr***:***expr*	conditional expression — evaluates first *expr* and if nonzero evaluates to the second *expr*; otherwise to the third *expr*
lvalue **^=** *expr*	raise *lvalue* to exponent *expr*
lvalue **%=** *expr*	assign *lvalue***%***expr* to *lvalue*
lvalue ***=** *expr*	multiply *lvalue* by *expr*
lvalue **/=** *expr*	divide *lvalue* by *expr*
lvalue **+=** *expr*	add *expr* to *lvalue*
lvalue **-=** *expr*	subtract *expr* from *lvalue*

continued...

Syntax	Description
lvalue **=** *expr*	assign *expr* to *lvalue*

All of the arithmetic operators are based on the C Standard. The conditional expression returns either strings or numbers, depending on the input expressions. Only one of the alternative expressions is evaluated.

In addition to the descriptions in the previous table, an expression can also be a floating-point number, a literal string enclosed by double quotation marks (**"**), or a variable name.

You can treat a variable or a field as a number or a string at any time, depending on its current usage. There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, you add zero to it. To force an expression to be treated as a string, concatenate the null string (**""**) to it. Variables and fields are set by the assignment statement:

*lvalue***=***expression*

The type of *expression* determines the resulting variable type.

The assignment includes these arithmetic assignments:

```
+=    -=    *=    /=    %=    ^=    ++    --
```

each of which produces a numeric result. The left-hand side of an assignment and the target of increment and decrement operators can be one of the following:

- a variable
- an array with index
- a field selector

This is indicated by the following BNF grammar:

```
<lvalue>:    <variable>
           | $expr
           | <variable> [ <index> ]
           ;
```

A valid array index consists of one or more comma-separated expressions, with one expression for each dimension of the array. Because **awk** arrays behave as associative memories, an array index can be any string. Since **awk** arrays are really one-dimensional, a multi-dimensional array index is converted to a one-dimensional index by concatenating all expressions, each separated from the other by the value of the *SUBSEP* variable (see Variables).

Thus, the following two index operations are equivalent:

```
var[expr1, expr2,... , exprn]
var[expr1 SUBSEP expr2 SUBSEP ... SUBSEP exprn]
```

A multi-dimensioned index used with the **in** operator must be parenthesized. The **in** operator, which tests for the existence of a particular array element, doesn't cause that element to exist. But any other reference to a nonexistent array element automatically creates it.

Comparisons are made numerically if both operands are numeric; otherwise, operands are converted to strings as required and a string comparison is made.

In the previous table, "**awk** Expressions in Decreasing Precedence," operators of higher precedence are grouped before those of lower precedence. In expression evaluation, higher precedence operators are evaluated before lower precedence operators. All operators associate to the left except for the assignment operators, the conditional operator (**?:**), and the exponentiation operator (^). Because the concatenation operation is represented by adjacent expressions rather than an explicit operator, you often need to use parentheses to enforce the proper evaluation precedence.

Variables

You can use variables in an **awk** program by assigning to them. They don't need to be declared — uninitialized variables have the value of the empty string, which has a numeric value of zero. All variables, including fields, are treated as string variables unless they're used in a clearly numeric context.

Field variables are designated by a **$**, followed by a number or a numerical expression.

You can create new field variables by assigning a value to them. References to non-existent fields — i.e. fields after "**$(***NF***)**" — produce the null string. But, assigning to a non-existent field (e.g. **$(***NF***+2)=5)** increases the value of *NF*, creates any intervening fields with the null string as their values, and causes the value of $*0* to be recomputed, with the fields being separated by the value of *OFS*.

Other special variables set by **awk** are shown in the following table:

Variable	Meaning
ARGC	the number of elements in the *ARGV* array
ARGV	array of command-line arguments — excluding options and the *program* argument — numbered from zero to *ARGC*-1
FILENAME	pathname of the current input file
FNR	ordinal number of the current record in the current file
FS	input field separator regular expression; **space** by default
NF	number of fields in the current record
NR	ordinal number of the current record from the start of input

continued...

© 2005, QNX Software Systems

Variable	Meaning
OFMT	print statement output format for numbers; **"%.6g"** by default
OFS	print statement output field separation; **space** by default.
ORS	print statement output record separator; **newline** by default
RLENGTH	length of string matched by the match function
RS	the first character of the string value of *RS* is the input record separator; **newline** by default. If *RS* is null, records are separated by blank lines, and **newline** is always a field separator, regardless of the value of *FS*
RSTART	starting position of string matched by *match* function, numbering from 1. This is always equivalent to the return value of the *match* function
SUBSEP	subscript separator string for multi-dimensional arrays; the default value is \034

You can modify or add to the arguments in *ARGV*; you can alter *ARGC*. As each input file ends, **awk** treats the text non-null element of *ARGV*, up through the current value of *ARGC*-1, as the name of the next input file. Thus, setting an element of *ARGV* to null means that it isn't treated as an input file. A dash (**-**) filename indicates the standard input. If an argument contains an equals sign (**=**), this argument is treated as an assignment rather than as a *file* argument.

Patterns

The structure of a pattern is specified by the following BNF grammar:

<*pattern***>**: **BEGIN**
| END
| <*simple pattern***>**

| *<simple pattern>*, *<simple pattern>*

;

<simple pattern>:

<simple expression>

| /*<ere>*/

;

In other words, a *pattern* is any valid *expression*, or an extended regular expression. In addition, a *pattern* can be a range specified by two of these patterns separated by a comma, or can be one of the two special patterns **BEGIN** or **END**.

Special patterns — BEGIN and END

The **awk** utility recognizes two special patterns, **BEGIN** and **END**. **BEGIN** is matched once and its associated action is executed before the first line of input is read and before command-line assignment is done. **END** is matched once and its associated action is executed after the last line of input has been read. These two patterns have associated actions.

BEGIN and **END** don't combine with other patterns. Multiple **BEGIN** and **END** patterns are allowed. The actions associated with the **BEGIN** patterns are executed in the order specified in the program, as are the **END** actions. An **END** pattern can precede a **BEGIN** pattern in a program.

If a program consists of:	Then:
only **BEGIN** blocks	**awk** exits without reading its input when the last statement in the **BEGIN** block is executed.

continued...

If a program consists of:	Then:
only **END** blocks or only **BEGIN** and **END** blocks	the input is read before the statements in the **END** block(s) are executed.

Regular expressions

The **awk** utility makes use of the extended expression notation, except that it allows the use of C language conventions for escaping special characters within the extended regular expressions, as you can see in the following table:

Escape	Meaning
\b	**backspace**
\f	**form-feed**
\n	**newline**
\r	**carriage return**
\t	**tab**
ddd	1-3 digit octal value *ddd*

If *ere* is an extended regular expression, the pattern:

/ere/
matches any line of input that contains a substring specified by the regular expression. You can limit a regular expression comparison to a specific field or string by using one of the two regular expression matching operators, **~** and **!~**. For example:

```
$4 ~ /ere/
```

matches any line in which the fourth field matches the regular expression */ere/*.

This pattern:

```
$4 !~ /ere/
```

matches any line in which the fourth field doesn't match the regular expression **/***ere***/**.

You can use an extended regular expression to separate fields by using the **-F** *ere* option, or by assigning the expression to the built-in variable *FS*. The default field separator is a single space character. The following describes the behavior of *FS*:

1 If *FS* is a single character:

- **1a** If *FS* is **space**, skip leading and trailing blanks; fields are delimited by sets of one or more blanks.
- **1b** If *FS* is any other character *c*, fields are delimited by each single occurrence of *c*.

2 Otherwise, if *FS* is more than one character, *FS* is considered to be an extended regular expression. Each occurrence of the string matching the extended regular expression delimits fields.

Pattern ranges

A pattern range consists of two patterns separated by a comma; in this case, the action is performed for all lines between an occurrence of the first pattern and the following occurrence of the second pattern, inclusive. At this point, the pattern range can be repeated starting at input lines subsequent to the end of the matched range.

Expression patterns

An expression pattern is considered to match — or be true — when the expression evaluates to a nonzero numeric value. Otherwise, the pattern is considered false.

Actions

An action is a sequence of statements. A statement can be one of the statements listed as follows. In this list, optional elements are shown in square brackets (**[ ]**) and keywords are shown in a **constant-width** typeface.

if (*expression* **)** *statement* [**else** *statement*]
while (*expression* **)** *statement*
do *statement* **while (** *expression* **)**
for ([*expression*]**;**[*expression*]**;** [*expression*]**)** *statement*
for (*variable* **in** *array* **)** *statement*
delete *array*[*index*]
break
continue
{ [*statement*]... **}**
variable **=** *expression*
next
exit [*expression*]
return [*expression*]
print [**(**] [*expression-list*] [**)**] [*redirection-expression*]
printf [**(**] *format*[**,** *expression-list*] [**)**] [*redirection-expression*]

Any single statement can be replaced by a statement list enclosed in braces (i.e. **{}**). The statements in a statement list are separated by **newline** characters or semicolons. The symbol **#** anywhere in a program line — in strings or EREs — begins a comment that is terminated by the end of the line.

Statements are terminated by semicolons or **newline** characters. You can split a long statement across several lines by ending each partial line with a backslash; **newline** characters without backslashes can follow:

- a comma
- an open brace
- a logical AND operator (**&&**)
- a logical OR operator (**||**)
- the **do** keyword
- the **else** keyword

- the closing parenthesis of an **if**, **for**, or **while** statement

For example:

```
{ print $1,
        $2 }
```

String constants are surrounded by double quotes (**"***string***"**). A string expression is created by concatenating constants, variables, field names, array elements, functions, and other expressions.

The *expression* acting as the conditional in an **if** statement is evaluated, and if it is nonzero and non-null, the next statement is executed. Otherwise, if *else* is present, the statement following the *else* is executed.

The **while**, **do...while**, **for**, **break**, and **continue** statements are based on the C Standard, except in the case of **for (** *variable* **in** *array* **)**, which iterates assigning each *index* of *array* to *variable* in order. The **for** statement has a form that processes each element in an array. The order of processing is unspecified.

The **awk** language supplies arrays that are used for storing numbers or strings. Arrays don't have to be declared, and their sizes change dynamically. The subscripts, or element identifiers, are strings that provide a type of associative array capability. Subscripts can't themselves be arrays.

The **delete** statement removes an individual array element. Thus, the following code deletes an entire array:

```
for (index in array)
        delete array[index]
```

The **next** statement causes all further processing of the current input line to be abandoned.

The **exit** statement invokes all **END** actions in the order in which they occur in the program source. A **next** statement inside an **END** also terminates the program and can optionally set the utility's exit status.

Output statements

Both **print** and **printf** statements send their output to standard output by default. The output is written to the location specified by *redirection-expression*, if one is supplied, as follows:

>*expression*
>>*expression*
|*expression*

In all cases, the *expression* is evaluated to produce a string that's used as a full pathname to write into (for **>** or **>>**) or as a command to be executed (for **|**). Using the first two forms, if the file of that name isn't currently open, it is created if necessary, opened, and using the first form, truncated. The output is then appended to the file. Subsequent calls in which *expression* evaluates to the same name simply append output to the file; the file remains open until closed.

The third form writes output onto a stream compatible with *popen()*. If no stream is currently open, the stream is created with the same command name; the stream created is compatible with *popen()* invoked with a mode of **w**. Subsequent calls write output to the existing stream if, in those calls, expression evaluates to the same command name as a stream that's currently open. The stream is closed as if *pclose()* were called with an expression that evaluates to the same command name.

The **print** statement writes the value of each expression argument to the indicated output stream, separated by the current output field separator (see *OFS* in the table "**awk** Variables"), and terminated by the output record separator (see *ORS* in the table). String *expressions* are written out as is; numeric *expressions* are written out as if produced by **printf** using a format that is the string value of the variable *OFMT*. The *expression-list* is a comma-separated list of *expressions*. An empty *expression-list* stands for the whole input line ($*0*).

With **printf**, the expressions are printed according to the specified format. A format argument is required — all other arguments in

expression-list are optional. The string value of the expression *format* is interpreted in a manner similar to the C function *printf()*, as follows. In the *format* string, format specifications begin with the single character %, and can optionally include the following three modifiers:

Modifier	Meaning
-	left justify the expression in its field
width	pad left to this width as needed; a leading 0 pads with zeros
.prec	maximum string width, or digits to right of decimal point

Format specifications are terminated by any other character. For each format specification that consumes an argument, the next argument from *expression-list* is evaluated and converted to the appropriate type (string, integer, or floating point). Both **`print`** and **`printf`** can output at least 1024 bytes.

Here are the format-specification characters:

Character	Interpretation
c	if the argument is numeric, print a character; if the argument is a string, print only the first character
d	decimal integer
e	exponential notation: [**`-`**]*d.dddddd***`E`**[**`+-`**]*dd*
f	floating point: [**`-`**]*ddd.dddddd*
g	shorter of e or f notations; suppress nonsignificant zeros
o	unsigned octal number

continued...

Character	Interpretation
s	string
x	unsigned hexadecimal number
%	print a %; no argument consumed

Functions:

The **awk** language has a variety of built-in functions — arithmetic, string, input/output, and general.

Arithmetic Functions:

The arithmetic functions, except for *int*, are based on the C Standard.

atan2(*y*,*x*)	returns arctangent of *y*/*x*
cos(*x*)	returns cosine of *x*, where *x* is in radians
sin(*x*)	returns sine of *x*, where *x* is in radians
exp(*x*)	returns the exponential function of *x*
log(*x*)	returns the natural logarithm of *x*
sqrt(*x*)	returns the square root of *x*
int(*x*)	truncates its argument to an integer; truncated toward 0 when *x***>**0.
rand()	returns a random number *n*, such that 0**<**=*n***<**1.
srand([*expr*])	sets the seed value for *rand()* to *expr* or uses the time of day if *expr* is omitted. The previous seed value is returned.

String functions:

gsub(*ere*,*repl*[,*in*])

Behaves like sub (see below), except that it replaces all occurrences of the regular expression in $*0* or in the *in* argument, when specified.

index(*s*, *t*) Returns the position, in characters, numbering from 1, in string *s* where string *t* first occurs, or zero if it doesn't occur at all.

length[(*s*)] Returns the length, in characters, of its argument taken as a string, or of the whole, $*0*, if there is no argument.

match(*s*, *ere*) Returns the position, in characters, numbering from 1, in string *s* where the extended regular expression *ere* occurs, or zero if it doesn't occur at all. *RSTART* is set to the starting position (which is the same as the returned value), zero if no match is found; *RLENGTH* is set to the length of the matched string, -1 if no match is found.

split(*s*, *a*[, *fs*]) Splits the string *s* into array elements *a*[1], *a*[2],..., *a*[*n*], and returns *n*. The separation is done with the extended regular expression *fs* or with the field separator *FS* if *fs* isn't given.

sprintf(*fmt*, *expr*, *expr*, ...)

Formats the expressions according to the printf format given by *fmt* and returns the resulting string.

sub(*ere*, *repl*[, *in*])

Substitutes the string *repl* in place of the first instance of the extended regular expression *ere* in string *in* and returns the number of substitutions. If *in* is omitted, **awk** uses the current record ($*0*).

substr(*s*, *m*[, *n*]) Returns the *n*-character substring of *s* that begins at position *m*, numbering from 1. If *n* is missing, the

length of the substring is limited by the length of the string *s*.

All of the preceding functions that take *ere* as a parameter expect a pattern or a string valued expression that is a regular expression.

Input/Output and general functions:

close(*expression*)

Closes the file or pipe named *expression*.

expression **|** *getline* [*var*]

Pipes the output of the command string given by expression into *getline*; each successive call to *getline* returns the next line of output from *expression*. This construct has the behavior of *popen()* called with a *mode* of **r**. If *var* isn't specified, $*0* and *NF* are set; if *var* is specified, *var* is set.

getline Sets $*0* to the next input record from the current input file. This form of *getline* sets *NF*, *NR*, and *FNR* variables.

getline **<** *expression*

Sets $*0* to the next record from the pathname expression. This form of *getline* sets the *NF* variable.

getline var Sets variable *var* to the next input record from the current input file. This form of *getline* sets *FNR* and *NR* variables.

getline var **<** *expression*

Sets *var* from the next record of expression, treated as a pathname. This form of *getline* doesn't set the *NF*, *NR*, and *NFR* variables.

system (*expression*)

Executes the command given by *expression* in a manner consistent with the *system()* function and returns the exit status.

All forms of *getline* return 1 for successful input, zero for end of file, and -1 for an error.

User-defined functions

The **awk** language also provides user-defined functions. You can define such functions — in the pattern position of a pattern-action statement — as:

function *name*(*args*,...) {*statements*}

A function can be referred to anywhere in an **awk** program. In particular, a function call can precede its definition. The scope of a function is global.

Function arguments are passed by value if scalar and by reference if an array name. Argument names are local to the function; all other variable names are global. The number of parameters in the function definition doesn't have to match the number of parameters in the function call. Excess formal parameters can be used as local variables. If fewer arguments are supplied in a function call than are in the function definition, the extra receiving parameters are left uninitialized.

When you're invoking a function, remember that no white space is allowed between the function name and the opening parenthesis. Function calls can be nested and can be recursive. You can use the **return** statement to return a value.

In the function definition, **newline**s are optional before the opening brace and after the closing brace. Function definitions can appear anywhere in the program where a pattern-action statement is allowed. In a function call, no white space is allowed between the function name and the opening parenthesis that begins the function parameter list.

Sample awk programs:

Note that the following are sample **awk** programs, not complete command lines.

Write to the standard output all input lines for which field 3 is greater than 5:

```
$3 > 5
```

Print every tenth line:

```
(NR % 10) == 0
```

Print any line with a substring that matches the regular expression:

```
/(G|D) (2[0-9][[:alpha:]]*)/
```

Print the second to last field and the last field in each line; separate the fields by a colon:

```
{OFS=":";print $(NF-1), $NF}
```

Print the line number and number of fields in each line. The three strings representing the line number, the colon, and the number of fields are concatenated and the resulting string is written to standard output:

```
{print NR ":" NF}
```

Print lines that are longer than 72 characters:

```
length $0 > 72
```

Print the first two fields in opposite order separated by the *OFS*:

```
{ print $2, $1 }
```

Same as above, with input fields separated by a comma or **space** and **tab** characters, or a combination of all these:

```
BEGIN  {FS =  ",[ \t]*|[ \t]+ " }
       { print $2, $1 }
```

Add up the first column, and print the sum and the average:

```
    {s += $1 }
END {print  "sum is  ", s,  " average is ", s/NR}
```

Print the fields in reverse order, one field per line (i.e. many lines out for each line in):

```
{ for (i = NF; i > 0; --i) print $i }
```

Print all lines between occurrences of the strings start and stop:

```
/start/, /stop/
```

Print all lines whose first field is different from the first field of the previous line:

```
$1 != prev { print; prev = $1 }
```

Simulate echo:

```
BEGIN {
  for (i = 1; i < ARGC; ++i)
    printf  "%s%s ", ARGV[i], i==ARGC-1? "\n ": "  "
}
```

If there's a file named **myfile** that contains page headers of the form:

```
Page #
```

and a file named **program** that contains:

```
/Page/{ $2 = n++; }
{ print }
```

then the command line:

```
awk -f program n=5 myfile
```

would print the file **myfile**, filling in page numbers starting at 5.

Examples:

Print the file **myfile**, which contains page references, filling in page numbers starting at 5:

```
awk '/Page/{ $2=n++; } { print }' n=5 myfile
```

Files:

By default, input files are text files that are read in order. You can modify either variable *ARGV* or variable *ARGC* to place this default file processing under program control.

The nature of the output files depends on your **awk** program.

Environment variables:

PATH Defines the search path when looking for commands executed by *system*(*expr*), or input and output pipes.

Exit status:

0 All input files were processed successfully.

>0 An error occurred.

Contributing author:

GNU

See also:

Dale Dougherty, *sed & awk*, O'Reilly and Associates, 1990.

A.V. Aho, Brian W. Kernighan, and Peter J. Weinberger, *The AWK Programming Language*, Addison-Wesley, 1988.

© 2005, QNX Software Systems

basename

Return nondirectory portion of pathname (POSIX)

Syntax:

basename *string* **[***suffix***]**

Options:

string A string of text.

suffix A string of text.

Description:

The **basename** utility is useful primarily for extracting the filename portion of a pathname, but since it performs only string operations, it can be used with any string.

The **basename** utility prints to standard output a substring of its *string* argument, plus a **newline** character. The **basename** utility forms this substring by doing the following, in order:

1. discarding any trailing slash (**/**) characters
2. deleting from the pathname all characters up to and including the last slash character in the *string* argument
3. deleting the *string* argument's suffix, provided you've specified a *suffix* operand that's identical to the *string* operand's suffix.

   If *suffix* is equal to the remaining string, the suffix won't be removed (e.g. if *suffix* is **prog.c** and the remaining string is **prog.c**, the suffix **.c** isn't removed).

The result may be a null string only if *string* is a null string (**""**). In this case, **basename** outputs a single **newline** character.

If *string* consists entirely of slash characters, **basename** prints a single slash, followed by a **newline** character.

You'll use the **basename** utility most often within shell scripts, where it's normally invoked inside back-tick (‘...‘), or contained in **$(**...**)**.

© 2005, QNX Software Systems

Examples:

Command:	Output:
`basename .`	`.`
`basename /usr/src/prog.c`	`prog.c`
`basename /usr/src/prog.c .c`	`prog`
`basename /usr/src/prog.c .a`	`prog.c`
`basename /usr/src/`	`src`
`basename ...//[fred]`	`[fred]`

Files:

The **basename** utility writes the basename string derived from the command-line parameters to its standard output.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

awk, **dirname**, **fullpath**

Syntax:

bc [-l] [*file***...]**

Options:

-l ("el") Include a library of defined math functions and set the scale to 20. (See "Library functions," below.)

file The pathname of a text file containing commands and function definitions to be interpreted by **bc**. If any files are specified, **bc** processes those files, then reads from the standard input.

Description:

The **bc** utility is an interactive, programmable calculator that supports a complete set of control structures, including functions.

The **bc** utility supports 26 functions, 26 simple variables, and 26 array variables — each array may have up to 2048 elements. In addition, the utility performs arithmetic operations using a radix 100 number system with user-definable precision. When you specify the precision, the numbers are *exact* to that precision, unlike binary floating-point representation where rounding errors may compromise accuracy. The utility also operates in different bases, so you can easily convert numbers from one base to another.

Many common programming language constructs are supported, including:

- **if**, **while**, and **for** statements
- user-defined functions with parameters
- local variables

The **bc** utility provides no support for character or string manipulation. The syntax of **bc** is derived from C, but doesn't constitute a programming language.

Let's look at a very simple **bc** program:

```
"hello, world\n"
```

When this program is run, the statement **"hello, world"** is echoed with a **newline** character. In general, the result of any expression that isn't assigned to a variable or used in a control structure will be echoed. For example, the following statement:

```
5^2
```

will cause **bc** to respond with **25**

Note that the caret (**^**) is an integer exponentiation operator: *a*^*b* is *a* raised to the *b*th power, where *b* is truncated to an integer in the range -2^31 to +2^31. All of the "usual" arithmetic operators (**+**, **-**,*****,**/**,**%**) are available, but **%** is the remainder, not the modulus.

Bases

Two special built-in variables let you choose the base in which numbers are input and output:

ibase specifies input base

obase specifies output base

When numbers are recognized, the input base determines the significance of each digit. For example, the value "66" in base 10 is:

```
    6 × 10^0 =  6
+   6 × 10^1 = 60
               --
               66
```

whereas in base 8 it is:

```
    6 × 8^0 =  6
+   6 × 8^1 = 48
              --
              54
```

Note that hex numbers are recognized in the different bases and that their values also change with the base. For example, the number **FF** is valid in base 10 and has the decimal value 165:

```
    F(=15) × 10^0 =  15
+   F(=15) × 10^1 = 150
                    ---
                    165
```

Setting the output base results in one of two numerical formats. For bases in the range 2 to 16, the format consists of the alphabet 0-9,A-F. For bases in the range 17 to 10000, the format consists of a series of decimal digits with groups separated by spaces. In this format, each group represents a digit of significance, thus the number 61 in **obase= 20** is printed as:

```
   3 1
```

which should be interpreted as:

```
    1 × 20^0 =  1
+   3 × 20^1 = 60
               --
               61
```

Variables

The **bc** utility supports 26 simple variables, named *a* through *z* (case is significant), and 26 arrays, also named *a* through *z*. The context determines whether the simple variable or the array is selected. All variables are auto-initialized to 0 and are always available (i.e. there's no declaration statement).

Functions may create local variables (with the **auto** statement) that hide the global variables of the same name while the function is executing. Arrays are limited to 2048 elements. Variables may be referenced (right valued) or assigned to (left valued).

© 2005, QNX Software Systems

Assignment operators

The assignment operator has many variations, as in the C language. For example:

```
a += x
```

means the same as:

```
a = a + x
```

All of the binary operators have a corresponding assignment operator.

An assignment expression (i.e. **a=10**) has a "right value" that is the new value of *a*. Thus:

```
a = b = 10
```

assigns 10 to b, then assigns b to a.

The increment and decrement operators may be applied only to variables. The statement:

```
b = a++
```

is semantically the same as:

```
b=a; a = a + 1;
```

and the statement:

```
b = ++a
```

is semantically the same as:

```
a=a+1; b = a;
```

The following table summarizes operator precedence and associativity:

Operator	Associativity	Class
++, –	nonassociative	(increment, decrement)
-	right to left	(unary minus)
^	right to left	(exponentiation)
*, /, %	left to right	(multiplicative)
+, -	left to right	(additive)
==, **<**=, **>**=, !=, **<**, **>**	left to right	(comparative)
= +=, -=, *=, /=, %=, ^=,	right to left	(assignment)

The if statement

The **if** statement takes the following form:

if (*expr***)** *statement*

The *statement* is executed only if *expr* evaluates to a nonzero quantity. Regardless of the value of *expr*, the next statement is executed.

```
if (i > 2047) {
    "i exceeds array limit"
    i = 2047;         /* limit index */
}
a[i]      /* always print value */
```

Iteration statements

The **bc** utility supports two iteration statements: **while** and **for**. In both of these structures, **break** means “exit the loop immediately,” and **continue** means “pretend you have finished executing *statement*.”

The **while** statement has the general form:

while (*expr***)** *statement*

and is executed following this algorithm:

1. If *expr* is false, goto 4
2. execute *statement*
3. goto 1
4. proceed

In the **while** statement, **continue** and **break** are analogous to goto 1 and goto 4, respectively.

The **for** statement has the general form:

for (*expr1***;***expr2***;***expr3***)** *statement*

and is executed following this algorithm:

1. *expr1*
2. if *expr2* is false, goto 6
3. execute *statement*
4. execute *expr3*
5. goto 2
6. proceed

In the **for** statement, **continue** and **break** are analogous to goto 4 and goto 6, respectively.

User-defined functions

In **bc** you can define named functions that support parameters, local variables, and recursion. A function name is a single lowercase character (a to z). A function has the general form:

define <*name***>(***parameter-list***) {** *function-body* **}**

You cannot remove or replace functions; it is an error to attempt to do so. Functions can call other functions, including themselves.

Functions can accept an arbitrary number of parameters, which may be either simple or array variables.

Parameters are passed by value, that is, the function is given a private copy of all parameters that remains in effect until the function returns. Notice that arrays are also passed by value — this differs from the C calling convention.

The first statement in a function can be an auto statement, which creates "local" variables for the function. The auto statement has this general form:

auto [*variable*|*array*]...

The local variables are used in place of the global variables of the same name until the function returns; the local variables are also initialized to zero (see example below).

Functions can contain a **return** statement. The **return** statement causes the function to exit, discarding any local variables and parameters. The statement may be in either of two forms:

return

or

return [*retval*]

where *retval* is a constant or a variable name. If no specific **return** statement is given, or if *retval* is omitted, a value of 0 will be returned.

The following example shows various constructs being used in functions:

```
/* function s(b[], n)
 * return the sum of the first N elements of b[]
```

```
 */
define s(b[],n) {
auto t,i;     /* note that the ';'s are for style,
                 and are not required. */

    if (n > 2048) {
        "Array length out of bounds...\n"
    return 0;
    }
    for (i=0; i < n; i++) {
        t += b[i];
    }
    return t;
}
```

The function *s()* is introduced by the **define** statement, which also reveals the parameter types. It is important that all functions be called with the appropriate types and number of parameters. Failure to do so will result in unpredictable errors.

All parameters in **bc** are passed "by value," including arrays. This is unlike C, where arrays are always passed by reference.

The auto statement introduces the locally scoped variables *t* and *i*. Automatic variables are used to prevent unwanted side effects of using global variables.

Built-in variables and functions

Several built-in control variables affect how **bc** operates. The following variables may be set or queried:

Name	Function	Range	Default
scale	minimum precision maintained in calculations	0..100	0
ibase	set the radix for input	2..10000 numbers	10
obase	set the radix for output	2..10000 numbers	10

The **bc** utility provides the following built-in functions:

Function	Returns
sqrt(*expression*)	square root of the value
length(*expression*)	number of significant decimal digits in the expression
scale(*expression*)	number of decimal digits to the right of the decimal marker

Library functions

The **bc** library is enabled if you invoke **bc** with the **-l** option, or if you include the interpretation file **/usr/lib/bclib.b** in the command line. The library sets the scale to 20, which can be overridden by assigning the scale a new value. The following functions are defined in the library:

Syntax	Function
a(*expression*)	arctangent (in radians)
c(*expression*)	cosine (in radians)
e(*expression*)	exponential function
k(*expr1*,*expr2*)	Bessel function of integer order
l(*expression*)	natural logarithm
s(*expression*)	sine (radians)

Files:

Input files (either standard input or command-line *file* parameters) must be ASCII text files containing sequences of **bc** statements that are to be executed as they are read.

© 2005, QNX Software Systems

The file **/usr/lib/bclib.b** contains the **bc** library, a set of function definitions which are loaded automatically when the **-l** option is specified. This file must be present for the **-l** option to work.

Exit status:

0 Successful completion.

>0 An error occurred.

Caveats:

The **bc** utility's syntax contains a few ambiguities, which are outlined here.

The notion of a *statement* in **bc** is either a structured statement or an expression followed by a **newline** or a semicolon (**;**). An expression may be nil, thus a single **newline** forms a statement. This can lead to unexpected behavior, as in the following example, where the **while** statement never terminates:

```
a=0
while (a < 100)
    a++
a
```

Although the intent is obvious, the **newline** at the end of the **while** statement comprises the body of the loop, thus *a* is never incremented. The C equivalent of this example would be:

```
a=0;
while (a < 100);
```

The following are "correct" versions:

```
a=0
while (a < 100) a++
```

or

```
while (++a < 100)
```

or

```
while (a < 100) {
    a++
}
```

The **quit** statement causes **bc** to exit when it encounters the statement, not when it attempts to execute it. Thus, the following will cause **bc** to exit immediately:

```
if (0) {
    quit
}
```

and is equivalent to:

```
quit
```

See also:

Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978.

Blkfsys

© 2005, QNX Software Systems

Block special filesystem (QNX)

Syntax:

Blkfsys [-b *num*[**,***size*]**] [-f** *ocb*] **[-N** *name*]
[-T] [-v *level*] **&**

Options:

-b *num*[,*size*]	Set the number of buffers and, optionally, the size of each buffer (in 512-byte blocks). The default is 1,64. If the device is read-only or is a tape, this is all the buffer space you will need. If you are writing to disks (e.g. **vol** to a floppy) then specifying more buffers will speed up operation.
-f *ocb*	Set the maximum number of control blocks that may be open concurrently. The default is 64.
-N *name*	Specify the registered name to be attached so drivers can locate the manager. The default is **qnx/fsys32**.
-T	Disable threaded operation in driver.
-v *level*	Set the verbosity level. The default is 1.

Description:

The Block Filesystem Manager (**Blkfsys**) is used to provide access to devices supported by available **Fsys** drivers as block special files, where the devices will not need to be mounted as a filesystem. While **Fsys** also provides this function, it does so at a significant cost in memory consumed.

Blkfsys is normally used in situations where a tape drive, floppy, or CD-ROM is present on a machine and a normal QNX filesystem will not be present on those devices, or in situations where a machine has a hard drive but the drive does not have a QNX filesystem on it (e.g. a hard drive with only DOS on it; **Dosfsys** would be used in conjunction with **Blkfsys**).

Examples:

Start the block filesystem with default parameters.

```
Blkfsys &
```

Start the block filesystem with four 32k buffers.

```
Blkfsys -b 4 &
```

Exit status:

Blkfsys will not exit if started successfully.

>0 **Blkfsys** failed to start. The reason will be stated in a diagnostic message written to standard error.

See also:

Dosfsys, **Fsys**, **Fsys.***, **Iso9660fsys**, **pax**

boot

© 2005, QNX Software Systems

Bootstrap the operating system

Syntax:

boot [-AEIOPV] [-dfptv] [-b *baud***] [-i** *ioport***] [-g** *n***] [-M** *addr*,*nbytes***] [-R** *n***]**

Options:

-A Don't probe BIOS for APM (Advanced Power Management).

-b *baud* Direct any output to a serial port set to the specified baud rate. Unless **-i** *ioport* is also specified, COM1 will be used.

-d Don't reprogram the serial controller — use it as is.

-E Emulate floating point. Tell QNX there's no floating-point unit available.

-f Don't turn off floppy motor.

-g *n* Adjust A20 gate as follows:

- **-1** Don't adjust — just ensure it's enabled.
- **1** Try "fast" gate before "slow" gate.
- **0** Autodetect (the default).

-I Don't enable interrupts when probing the machine.

-i *ioport* Direct any output to the serial port at the specified I/O address (0x*ioport*). Unless **-b** *baud* is specified, the serial port will be set to 9600 baud.

-M *addr*,*nbytes*

Add memory to list for QNX; *addr* is the start address, *nbytes* is the number of bytes. Both must be 4K aligned.

-O For old-style BIOSes. Disable APM and PnP (Plug'n'Play) memory probe.

-P	Disable PCI interface (int 0x1a).
-p	Pause before starting QNX. This is a debugging option of value only to check the names of boot modules printed by **-v**.
-R *n*	Reserve the first megabyte plus *n* 64K chunks of memory.
-t	Continue even if A20 gate fails.
-v	Verbose. Print information, including names of boot modules.
-V	Disable video.

© 2005, QNX Software Systems

☞ The following options are really intended to help you get around buggy BIOSes or hardware:

-A	Don't probe BIOS for APM.
-d	Don't reprogram the serial controller — use it as is.
-E	Emulate floating point.
-f	Don't turn off floppy motor.
-g *n*	Adjust A20 gate.
-I	Don't enable interrupts when probing the machine.
-O	For old-style BIOSes. Disable APM and PnP probe.
-P	Disable PCI interface (**int 0x1a**).
-t	Continue even if A20 gate fails.
-V	Disable video.

Description:

The **boot** program performs initial operations for the operating system. This includes operations which must be performed in "real mode," which is the startup mode of the Intel 386 family of processors. Before **boot** transfers control to QNX, the machine must be running in 32-bit protected mode.

☞ The **boot** program is used only within a boot image. If an operating system buildfile (see **buildqnx**) does not specify the **boot** program, **buildqnx** will automatically insert it (with no options). If it is explicitly specified, it must be the first entry in the buildfile.

The **boot** command uses the ISA-BIOS and some test code to detect some elements of the machine environment including bus-architecture (PCI, EISA, ISA, PS/2), processor (386,486,586...), numeric processor (287, 387,...), video-adapter (none, mono, CGA, EGA, VGA,...), and memory size.

Due to lack of foresight in the ISA-BIOS specification, the full memory size sometimes can't be identified. For this reason, **boot** provides an option to "add" memory to the system.

When **boot** has completed, it jumps to the entry point of the next program in the boot image. Conventionally, this is the process manager (**Proc32**). The process manager reclaims any memory used by **boot**.

Since **boot** is independent of the operating system, it is possible for other "boot" programs to be written for specialized hardware, possibly nullifying need of an ISA-BIOS.

If your target machine doesn't have a BIOS, you can use the **boot-nobios** utility. Its syntax and options are the same as **boot**'s.

Diagnostics:

If **boot** can't succeed, it prints:

```
BOOT ERROR:
message
--- cannot start operating system
```

where *message* can be one of:

no modules in image, check build file

The **boot** program was the only program in the boot image.

need 32-bit processor

Failed to detect a 32-bit CPU.

© 2005, QNX Software Systems

relocate, invalid BMCB
bmcb list invalid

The boot image is corrupt (Boot Module Control Block).

not enough memory

The system doesn't have enough memory to create startup tables.

run failed Catch-all; should never happen.

If **-v** is specified, **boot** prints the following just before jumping to the process manager:

boot modules:
list of subsequent modules in boot image
starting QNX...

Exit status:

The **boot** program does not exit in the normal sense. After the necessary setup has been performed it transfers control to the process manager (**Proc**) built into the boot image.

See also:

buildqnx, **Proc**

buildqnx

Build a custom version of QNX (QNX)

Syntax:

buildqnx [-b *baseaddr***] [-Vv] [***macro***=***name***]**
buildfile **[***os_file***]**

Options:

-b *baseaddr*	Load QNX into memory, starting at *baseaddr* (default is 0x1000). This is a linear address that must reside in the first 640K of memory. The address must be specified in hex, must be greater than or equal to 0x1000, and must be a multiple of 0x1000.
-v	Be verbose.
-V	Be very verbose.
*macro***=***name*	Define a macro.
buildfile	An input file of module names.
os_file	An output image file (default: standard output).

Description:

The **buildqnx** utility builds a QNX boot image. A boot image is a collection of processes that are "preloaded" into a file that's loaded into memory at boot time. Once the image is in memory, control is transferred to the first process in the image; this process starts each of the other processes. For the full 32-bit **Proc** (versions 4.23A and later), this control is first transferred to the code in **/boot/sys/boot**, which can be either explicitly named in the buildfile or implicitly added by **buildqnx** when it sees that the full 32-bit **Proc** is being used in the image. The **boot** code relocates the processes in the boot image, sets up segments, and then runs **Proc**.

You may create images statically by explicitly invoking **buildqnx**, or create them dynamically for machines booting from the network by running the **netboot** utility.

Each major QNX service is provided by a process. You can add or delete an operating system service by adding or deleting the process that manages it. Most processes may be started after the system boots and as such don't need to be included in an image.

For example, let's say you've booted from disk and the boot image didn't contain the DOS filesystem manager (**Dosfsys**). At this point you're getting your files and commands from a local QNX filesystem. You could start **Dosfsys** by typing its name on the command line and running it as a background process, as follows:

```
Dosfsys &
```

It's necessary to start some processes before the system becomes functional. For example, if a diskless workstation boots over the network, it needs to have a network manager in the image to access the remote filesystem. Likewise, if you boot from a hard disk, you can't start a filesystem manager process by loading it off the disk, since you don't yet have a filesystem to load from. In this case, build a boot image containing **Fsys**, and an Fsys driver, and **mount** so that the filesystem is available immediately upon loading.

As a general rule, you should build into the image only the minimum services needed, and start other services after booting. In some environments, you may want to freeze an OS and a larger number of services by building a large image that you don't change. If you're in a special embedded environment where dynamic loading isn't possible (such as would be the case if, for instance, neither a network manager (**Net**) nor filesystem manager (**Fsys**) were going to be run), you may also choose to build a complete load image of all the processes you need.

The core kernel modules and images are placed under the **/boot** directory. This directory contains:

Makefile	A file that references a set of buildfiles that name the processes to be included in an image.
sys	A directory that contains key image components that aren't runnable on their own from the command line,

such as **Proc**, **Slib16** and **Slib32**, **Debugger32**, and **boot**.

The following table lists standard QNX processes that are often included in an image:

Category	Process	Description
A	**boot**	Operating system bootstrap
B	**Proc**	Process manager, including the microkernel
B	**Slib32**	32-bit shared libraries
H	**Slib16**	16-bit shared libraries
C	**Fsys**	QNX filesystem manager
C	Fsys.* drivers	QNX filesystem drivers
D	**Dosfsys**	DOS filesystem manager (requires **Fsys**)
D	**Dev**	Device manager
D	Dev.* drivers	Device drivers
E	**Net**	Network manager
E	Net.* drivers	Network drivers
F	**sinit**	System initialization process
G	**Debugger32**	Low-level system debugger (not for general use)

The process categories are as follows:

A Must be present as the first entry in the image. If it's omitted from the buildfile, **buildqnx** automatically inserts it (without any options).

B Must be present in an image as the first two processes after **boot**.

C Needed if you want to boot from a disk controlled through that adapter type.

D Can be started later in the sysinit file — we recommend this method.

E Needed if you want to boot over the network; otherwise can be started in the sysinit file. (May be omitted when booting over the network if the machine being booted is a standalone machine that doesn't need to read the sysinit or fetch other resources at boot time from the boot server.)

F Normally included to execute a shell file of processes that you start dynamically.

G Needed for low-level system work.

H Required for running 16-bit executables from versions of QNX prior to 4.23.

You must include the Process manager (**Proc**) and the 32-bit shared library (**Slib32**) in the boot image. Each manager typically requires that you include one or more drivers if it's to be useful.

The operating system bootstrap, **boot**, is automatically inserted as the first entry in the boot image if it isn't explicitly specified. To access the optional functions of the bootstrap (such as its **-v** (verbose) option, which lists the components that are loaded in the boot image) you must explicitly specify **boot** as the first entry, supplying the desired options.

The list of programs to build into the image is kept in a buildfile under **/boot/build**. Each program occupies two lines in this file (empty lines are ignored). The first line is the pathname of the program. The second line starts with a **$** followed, in order, by:

1 An optional number representing the size of the initial heap (alloc space)

2 An optional :*nn*, where *nn* is the starting priority of the process

3 An optional comma (**,**) followed by one or more of the following modifiers:

- **d** or **D** Mark the process as the system debugger.
- **m** or **M** Mark the code as impure (i.e. copy it from XIP or ROM into RAM).

4 A command line.

If you don't specify a priority, your program starts at priority 10. A program always starts with a round-robin scheduling algorithm.

The following is a simple buildfile for creating an image to boot from a hard disk with an enhanced IDE controller on a single-node computer. The **sinit** utility is used to execute the system initialization file:

```
sys/Proc
$ Proc

sys/Slib32
$ Slib32

sys/Slib16
$ Slib16

/bin/Fsys
$ Fsys

/bin/Fsys.eide
$ Fsys.eide

/bin/mount
$ mount -p /dev/hd0 /dev/hd0t77 /

/bin/sinit
$ sinit TERM=qnx
```

© 2005, QNX Software Systems

Likewise, the following buildfile creates an image to boot a workstation over an Ethernet network using an NE1000/2000 compatible Ethernet card:

```
sys/Proc
$ Proc -l $(lnode)

sys/Slib32
$ Slib32

sys/Slib16
$ Slib16

/bin/Net
$ Net -m $(netmap)

/bin/Net.ether1000
$ Net.ether1000

/bin/sinit
$ sinit -r //$(bnode)/ TERM=qnx TZ=$(TZ)
```

In both the previous examples, the operating system bootstrap was inserted first into the boot image by **buildqnx**. The bootstrap may be specified explicitly, allowing options to be passed to the bootstrap. In this example, the bootstrap displays the OS components in the boot image as it executes:

```
sys/boot
$ boot -v

sys/Proc
$ Proc -l $(lnode)

sys/Slib32
$ Slib32

/bin/Net
$ Net -m $(netmap)

/bin/Net.ether1000
```

```
$ Net.ether1000

/bin/sinit
$ sinit -r //$(bnode)/ TERM=qnx TZ=$(TZ)
```

The following macros are replaced with appropriate information when **buildqnx** is run. The values used for the macro replacement must be specified via **buildqnx**'s command line arguments (*macro*=*name*). Normally this is done by the **netboot** utility when it invokes **buildqnx**.

This macro:	Is replaced by the:
$(lnode)	Workstation's logical node ID
$(bnode)	Boot server's filesystem root, under which **/etc/config/sysinit.***node* may be found
$(netmap)	Logical-to-physical node mapping of the boot server
$(TZ)	Time zone setting; used to allow booting nodes to come up automatically inheriting the timezone of the network boot server (macro must be set in **sinit**'s command line arguments).

Using the supplied Makefile

A makefile is provided in **/boot** as a convenience for building *static* boot images. It allows for images to be built based on macros that define disk drive types, etc, for which the makefile supplies default values if none are specified.

Buildfiles that use these macros can't be used for network booting because **netboot** doesn't set the macros when it runs **buildqnx**.

Let's say the hard disk buildfile is in **/boot/build/hard.1**. You could build an image with the **make** utility, using the following:

```
cd /boot
make b=hard.1
```

An image, called **hard.1**, is placed in the **/boot/images** directory. To make this image bootable from a hard disk, copy it to the **/.boot** file in the root directory of the hard disk. Before copying the file, you may wish to make a copy of the existing boot image — in case you made a mistake in your buildfile — by copying the existing **/.boot** to **/.altboot**, as follows:

```
cp /.boot /.altboot
cp /boot/images/hard.1 /.boot
```

We've provided a Makefile and a few standard buildfiles in the **/boot** directory that you can use as models:

To make a hard disk image:

make b=hard.1 d=*driver_name*

Starting processes omitted from the image

You should start the processes that were omitted from the image in the system initialization file, which is executed by the **sinit** process. This process attempts to execute the **/etc/config/sysinit.***node* file on the current node. If the file doesn't exist, **sinit** attempts to execute the **/etc/config/sysinit** file.

For more information, see the section on the system initialization file in the Basic Installation chapter of the QNX *Installation & Configuration* manual.

Examples:

Build the OS image **/boot/images/ws** from the buildfile **/boot/build/ws**:

```
cd /boot
buildqnx build/ws images/ws
```

Files:

The **buildqnx** command requires an ASCII text *buildfile* to be specified on the command line, and writes the resulting binary OS image to the *os_file* file named on the command line, or to standard output if no *os_file* is specified.

When the binary isn't being written to the standard output and **-v** or **-V** is specified, informative messages are written to the standard output as **buildqnx** does its work.

When **buildqnx** encounters an error, a diagnostic message is written to standard error.

The **buildqnx** command doesn't use standard input.

Exit status:

0 Successful.

>0 An error occurred.

Caveats:

The syntax of the buildfile is fragile. The **buildqnx** command "knows" that there are at least two characters (the newline and the **$**) between the tail of the actual file (e.g. **/bin/Fsys**) and the *argv*[0] string (e.g. **Fsys**). Intermediate chars are squeezed out and *argv*[0] is copied to one character after the tail of the filename.

By default, **buildqnx**:

- Inserts the command **sys/boot** if the first program isn't 16 bit.
- Marks ***Slib*** as impure code if it's a 16-bit program (i.e. the code must be copied from XIP or ROM into RAM).
- Marks ***Debugger*** as the system debugger and impure.
- Translates ***Debugger** to ***Debugger32**.

Also, a 32-bit program matching ***Slib*** defaults to priority 28, and a 16-bit program matching ***Slib*** defaults to priority 26. This is to ensure proper startup order.

© 2005, QNX Software Systems

We are quite embarassed.

See also:

boot, **netboot**, **sinit**, **tinit**

QNX *Installation & Configuration* guide

Syntax:

cat [-c] [-u] [*file***...]**

Options:

-c Compress each sequence of multiple **newline** characters into a single **newline** character.

-u Write unbuffered output. Data from the input file(s) is written to the standard output without delay as each file is read.

file The pathname of an input file. If no files are specified, the standard input is used. If a *file* is the dash character (**-**), **cat** reads from the standard input at that point in the sequence.

Description:

The **cat** utility reads files in the order you specify and writes the contents of these files to the standard output.

Examples:

Write the contents of the file **myfile** to the standard output:

```
cat myfile
```

Concatenate the files **doc1** and **doc2** and write the result to the **doc.all** file:

```
cat doc1 doc2 > doc.all
```

Append the contents of node 1's **/.licenses** file to node 5's **/.licenses** file:

```
cat //1/.licenses  >>//5/.licenses
```

© 2005, QNX Software Systems

Files:

The **cat** utility writes the contents of the files named on the command line (which may include the standard input) to the standard output.

If an error is encountered (e.g. in attempting to open or read a file), a diagnostic message will be written to the standard error.

Exit status:

0 All input files were output successfully.

>0 An error occurred.

Caveats:

Because of the shell language mechanism used to perform output redirection, a command such as:

```
cat doc doc.end > doc
```

causes the original data in **doc** to be lost. The file **doc** would be opened for write by the shell, and therefore truncated to zero length, before **cat** was executed.

See also:

cp, **head**, **tail**

cbpe

Use Byte-Pair Encoding to compress or decompress a file (QNX)

Syntax:

cbpe [options] [*srcfile* **[***dstfile***]]**

Targets:

QNX 4

Options:

-b *size*	Set the maximum block size before compression (default 4096).
-c *nchars*	Set the maximum character set before compression (default 200).
-d	Decompress the specified file. If this option is specified, all other options are ignored.
-f	Output as an embedded filesystem
-t *count*	Set the minimum threshold for byte pairs before replacement (default 3).

Description:

The **cbpe** utility compresses the file specified on the command line using a byte-pair encoding algorithm. This algorithm produces fair to good compression of most data. While the compression algorithm is relatively expensive to compute, the decompression algorithm is extremely fast, small, and simple. This allows it to be built directly into, for example, the embedded filesystem (Efsys) and the embedded system IPL for OS images (see **romqnx**).

By default, this utility always compresses. To decompress a file, specify the **-d** option.

Normally, the output is a file or character stream to the standard output. In the case of the embedded filesystem, you must inform the filesystem that this is a file that should be decompressed automatically

© 2005, QNX Software Systems

when read. To do this, specify the **-f** option; without it, the filesystem simply reads back the compressed data directly.

The **cbpe** utility reads the specified source file and compresses the data in blocks to the associated output file. During decompression, these blocks are re-expanded. To specify the maximum size a block can be expanded to when it's decompressed, use the **-b** option during compression.

A typical compressed file contains many back-to-back compressed blocks. A small block value creates overhead that reduces the compression ratio, while a block value larger than the default (4096) has little effect on increasing the compression factor and may in fact reduce it.

The compression algorithm looks for pairs of bytes that can be represented by a single byte. To do this, it needs to reserve some byte values for representing these pairs. This reduces the available character set of the source.

Input characters are read until either the block size has been reached or more unique characters have been read than the maximum character set permits. At this point a block of data is compressed. To change the maximum character set, use **-c**.

The default *count* value for **-t** provides reasonable compression for a wide range of output. But if the data has a limited character set, you may be able to save space by increasing the number. For data that contains all possible byte pairs, you may be able to save space by decreasing the number.

The decompressor doesn't need to know the maximum block size, character set, and minimum byte-pair values.

You should hold off on assigning a new code to byte pairs that may occur only once in the data. The default value prevents new codes from being consumed too easily. The total available number of new byte pairs is (256 - *nchars*), where *nchars* is set by **-c**.

Examples:

To compress the embedded shell and put it on an embedded filesystem (the shell will be decompressed by the filesystem when read):

```
cbpe -f esh /efs1p1/bin/esh
```

To compress a file called **data** and put it in the file called **data.bpe**:

```
cbpe data > data.bpe
```

or

```
cbpe data data.bpe
```

To decompress the file called **data.bpe** and put it in the file called **data**:

```
cbpe -d data.bpe > data
```

or

```
cbpe -d data.bpe data
```

See also:

romqnx

cd

© 2005, QNX Software Systems

Change working directory (POSIX)

Syntax:

cd [*directory***]**

Options:

directory The pathname of the directory that will become the new working directory. The interpretation of a relative pathname by **cd** depends on the **CDPATH** environment variable (see below).

Description:

The **cd** command is a shell built-in that changes the current or *working directory* of the shell to the directory named as its argument. All filenames that don't start with a slash (**/**) are assumed to be relative to your current working directory.

The command:

```
cd
```

with no arguments changes the current working directory to the directory named by your **HOME** environment variable set when you logged in. This is equivalent to:

```
cd $HOME
```

if the **HOME** environment variable is set.

If the **HOME** environment variable isn't defined, or has the null string as its value, then the current working directory is changed to the first non-NULL directory named in your **CDPATH** environment variable. If **CDPATH** is not set, the current working directory will not be changed.

The **cd** command makes use of **CDPATH**, which is analogous to the use of the **PATH** variable in the shell. It lets you name a directory with a one-component name rather than a full pathname.

CDPATH is a list of pathnames, separated by colons. If the *directory* specified in the **cd** command doesn't begin with a slash (**/**), and the first component isn't dot (**.**) or dot dot (**..**), **cd** searches for *directory* relative to each path named in the **CDPATH** variable, in the order listed. The new working directory is set to the first matching path found. An empty string in place of a directory pathname represents the current directory. If **CDPATH** isn't set, it is treated as if it were an empty string.

For example, if **CDPATH** is set to **/home/fred/:**, the command:

```
cd newdir
```

looks for the **/home/fred/newdir** and then for the **./newdir** directories and makes the first one of those it finds the working directory.

Some shells, including the standard QNX shell (**sh**), have extensions involving two argument invocations and use of **-** (dash) as an argument. See **sh** for details.

Symbolic links

In some UNIX systems, **cd** modifies the pathname given to it if that pathname contains symbolic links. As a result, the pathname of the new current working directory may differ from the one given to **cd**. In QNX, however, **cd** doesn't modify the pathname-aside from collapsing "**..**" references. For example:

```
cd /home/luc/test/../doc
```

would result in a current working directory of **/home/luc/doc**, even if some of the elements in the pathname were symbolic links.

For more information about symbolic links and "**..**" references, see *System Architecture*.

© 2005, QNX Software Systems

Examples:

Change directory to the directory named by the **HOME** environment variable:

```
cd
```

Change to the next highest directory in the directory structure:

```
cd ..
```

Change directory to the root directory:

```
cd /
```

Change to the **/path/dir** directory:

```
cd /path/dir
```

Change to the directory **/src**, resolved via the pathname space of node 61:

```
cd //61/src
```

Change to the directory two levels above the current working directory:

```
cd ../..
```

Change to the directory two levels above the current working directory, then to the **files** directory below in the directory tree:

```
cd ../../files
```

You can use the **pwd** command to display the current working directory.

Environment variables:

CDPATH When defined, **CDPATH** lists pathname components that are applied to any specified directory that doesn't begin with a slash (**/**), dot (**.**) or dot-dot (**..**).

HOME The name of the home directory, used when no *directory* operand is specified.

Exit status:

0 The directory was successfully changed.

>0 An error occurred; the directory remains unchanged.

Caveats:

The **cd** utility is available only as a shell builtin.

See also:

sh, **pwd**

System Architecture

cfont

© 2005, QNX Software Systems

Load text-mode font data into EGA or VGA cards (QNX)

Syntax:

cfont [-f *font_number***] [-r** *rows* **| -s** *scanlines***]** *font_file*

Options:

-f *font_number*

Load the font description into this font position (where *font_number* is a number from 0 to 3). If **-f** is omitted, the next free font slot is selected.

-r *rows* Support this number of screen rows on the console.

-s *scanlines* Support this number of scanlines for every character in the font.

font_file The name of a file containing a font description.

Description:

The **cfont** utility loads text-character font data into the text-mode console device driver for EGA or VGA display cards. Up to four fonts can be programmed into the driver, each occupying a font "slot" (*font_number*). Fonts can have a variety of typefaces, sizes, and styles (e.g. italics, bold, sanserif, upside-down), and using them may result in more than 25 active lines per screen. For **cfont** to work, the **Dev** manager and the console driver (**Dev.ansi** or **Dev.con**) must be running.

VGA character modes (with 400 scanlines per screen) support font sizes up to 16 scanlines per character, while EGA character modes (with 350 scanlines per screen) support font sizes up to 14 scanlines per character.

The **cfont** utility automatically determines how many scanlines per character the font definition describes by inspecting the size of the *font_file*. From this, the number of rows on the console is determined.

To override the default number of scanlines used by each character, use either **-r** or **-s**. For example, the shipped sanserif font is 14

scanlines high. Specifying **-s 15** would add an extra scanline to each row of characters, resulting in fewer rows being displayed. However, if **-s 13** was specified, the last scanline of each character would be discarded, resulting in more rows being displayed. Note that with many fonts the last scanline is white space, so decreasing the number of scanlines per character by 1 brings rows closer together but doesn't crop any characters.

The size of a font can also be changed with the **-r** option. Note, however, that it is an error to specify both **-r** and **-s** at the same time. The two options are related as follows:

On VGA:	On EGA:
rows = 400 ÷ *scanlines*	*rows* = 350 ÷ *scanlines*
scanlines = 400 ÷ *rows*	*scanlines* = 350 ÷ *rows*

Font files without absolute pathnames are searched for first in the current directory and then in the **/etc/config/cfont** directory. Several font varieties and sizes are included in this directory.

Examples:

Load the sanserif font description into system font position 0. This results in 28 lines per screen when displaying font 0 on a VGA monitor:

```
cfont -f0 sanserif.14
```

Files:

The **cfont** utility reads fonts from binary font files named on the command line. Font filenames which do not begin with a **/** will be looked for in the **/etc/config/cfont** directory.

© 2005, QNX Software Systems

Exit status:

0	font was loaded successfully
>0	an error occurred

See also:

Dev, **Dev.ansi**, **Dev.con**

chgrp

Change file group ownership (POSIX)

Syntax:

chgrp [-Rv] *group file*...

Options:

-R	Recursively change group ownership of files. For each *file* that names a directory, **chgrp** changes the group of the directory and of all files in the file hierarchy below it.
-v	Verbose. Display to standard output the operations which are being performed.
group	A group name from the group database, or a numeric group ID.
file	The pathname of a file whose group ID is to be modified.

Description:

The **chgrp** utility lets you change the group ownership of one or more files. For each file you name, **chgrp** sets the file's group ID to that specified by the *group* operand.

If you invoke **chgrp** with the **-R** option, and **chgrp** attempts but fails to change the group ID of a particular file in a specified file hierarchy, it continues to process the remaining files in the hierarchy. The **chgrp** utility can fail to change the group ID of a file if you don't have appropriate permissions.

☞ You must be **root** to change the group ownership from one group to another.

Normal users can change the ownership of a group *to themselves*, but they can't change the ownership of a group to which they don't belong.

Examples:

Change file group of **myfile** to 27:

```
chgrp 27 myfile
```

Change file group of **myfile** to **technical**:

```
chgrp technical myfile
```

Files:

/etc/groups

This file defines the known group IDs for the system. It associates each group name with a numerical group ID and a list of *username*s who are members of the group.

Entries in this file appear in the following format:

groupname::*groupid*:*user*[,*user*]...

Exit status:

0 The utility executed successfully and all requested changes were made.

>0 An error occurred.

See also:

chmod, **chown**, **find**

chkfsys

Check an entire QNX filesystem for consistency (QNX)

Syntax:

chkfsys [-PefpqrsuvV] [-z *zapfile***]** *drive*

Options:

-e	If possible, disable drive error correction during check for bad blocks.
-f	Don't fix anything.
-p	Prompt before starting.
-P	Prompt suppression (i.e. fix without asking any questions)
-q	Be quiet.
-r	Rebuild the bitmap without prompts or messages.
-s	Suppress statistics display.
-u	Check filesystem regardless of status recorded on disk.
-v	Be verbose. (Shows files in addition to directories as they are being checked. Slows **chkfsys** considerably.)
-V	Very verbose display.
-z *zapfile*	Record pathnames of files which need to be zapped in the specified file. The *zapfile* must reside on a different filesystem from the one being checked.
drive	The disk to check (e.g. **/dev/fd0** or **/dev/hd0t77** or **/**).

Description:

The **chkfsys** utility performs a consistency check of a QNX filesystem on the requested drive. The **chkfsys** utility will not operate on disk partitions containing non-QNX filesystems (e.g. DOS

partitions, QNX2 partitions). In addition, **chkfsys** must have access to the block special file that the filesystem is contained in. For this reason, **chkfsys** cannot be used on NFS-mounted QNX filesystems.

For QNX filesystems **chkfsys** recursively walks the filesystem, checking every file on the disk. During the walk, checks are made on the directory entry of each file and the extents that make up the file. A bitmap is constructed in memory that is consistent with the block allocation of all files and directories on the disk. This bitmap is then compared to the existing one on the disk. If they differ, the user is given the option of replacing the existing bitmap on disk with the one constructed in memory.

By default on systems running QNX 4.23 and later, **chkfsys** checks an on-disk flag which is maintained by the filesystem that indicates to **chkfsys** whether or not anything needs to be checked. If the flag is set, **chkfsys** reports that everything is fine and exits immediately. When you do an orderly shutdown of the system, this flag will always be set (unless an error had occurred in the process). If you shut down the system by powering down, the flag may or may not be set depending on the state of the filesystem at the time. You can force **chkfsys** to run despite the flag being set by using the **-u** option.

CAUTION:

The **chkfsys** utility should be used only when the filesystem is stable. There should be NO files open for writing when **chkfsys** is running.

In the normal mode, **chkfsys** will prevent its own operation when any files are open for writing on the drive. If **chkfsys** finds that no files are open for writing on the drive, it will proceed and will also prevent any files from being opened for writing on the drive for the duration of its check.

If you're not doing any fixes (with the **-f** option), you may check a filesystem with open files, but beware: you may get inconsistent reports in this case.

The **chkfsys** utility is normally used to recover blocks that were lost through the use of the **zap** utility. For example:

```
zap file1
```

In this case, **chkfsys** will report that there are blocks used in the bitmap that are in fact *not* used by any file. These blocks may be recovered by writing the reconstructed bitmap back to disk. The **chkfsys** utility will attempt to read each of these blocks, but will *not* mark bad blocks as available. Any blocks found this way will be added to the **/.bad_blks** file at the root of the filesystem being checked.

The **chkfsys** utility will tell you if any files are using blocks that are now known to be bad.

If **chkfsys** reports that a block is used by more than one file, this could indicate one of two problems:

- If bad blocks exist, then this means that the file uses a block that is bad and marked as used in the bitmap.
- If there are no known bad blocks, then a multiple allocation of a single block has occurred.

In either case, the file should be saved on *another* disk (if possible), and the original file should be destroyed with **zap**. The **chkfsys** utility should then be run again to update the bitmap, after which the saved file may be restored onto this disk.

In general, whenever the bitmap is replaced, **chkfsys** should be run a second time to ensure that the filesystem is indeed consistent. To do so you must specify the **-u** option.

The **-f** (no fix) option will prevent **chkfsys** from attempting to make any fixes to the filesystem. The disk will not be opened for write, but only for read. This option lets a user examine the filesystem without requiring other users to stop using the disk or filesystem. Beware, however, that the **-f** option may report errors that don't really exist (if other users are opening, closing, or growing files during the time that **chkfsys** is running). Even so, this can be a valuable option for sites

that are up and running 24 hours a day, when the system operator carefully evaluates the results. If you see errors that you believe result from current activity, run **chkfsys** again with the **-f** option to verify the errors. If you have located errors that require fixing, you should idle the filesystem and run **chkfsys** without the **-f** option.

The **-p** (pause) option is used primarily with floppy disks. You can start **chkfsys** from a floppy diskette, wait for **chkfsys** to pause, remove the current disk (which contains the **chkfsys** command), and then insert another disk you wish to check.

The **-q** (quiet) option suppresses the display of each filename as that file is checked. This will speed the checking significantly, without loss of information, because **chkfsys** will show you the name of any files that have errors.

The **-r** (rebuild) option suppresses the warning message that normally appears at the end of a **chkfsys** run when the existing bitmap differs from the newly constructed bitmap in memory. When **-r** is specified, **chkfsys** will automatically rebuild the bitmap. Note that this option is not effective with the **-f** (no fix) option.

The **-s** (no stats) option prevents the display of the statistics message that normally appears at the end of a **chkfsys** run.

The **-v** (verbose) option will display information on the checking.

The **-P** (no prompt) option will cause **chkfsys** to automatically fix problems encountered without prompting the user before each fix. Note that the **-P** option won't work with the **-f** option.

The **-z** *zapfile* option is used to record the names of files which should be zapped after **chkfsys** is finished to the named file, which must be on a different filesystem from the one being checked. When a file is found to use an area of the disk allocated to another file, or when a file uses an area of the disk marked as bad in the bitmap, the files must be zapped and **chkfsys** run again.

After a power failure

The **chkfsys** utility may also be run after a system crash or power failure, which may have left some files busy. The utility will make the files "unbusy" and will also make checks to ensure that no damage to the filesystem has occurred. QNX is designed to be immune to this type of damage.

In the event of the loss of a filesystem due to the corruption of the root directory and the bitmap (first few blocks of the disk), you should refer to the QNX *System Architecture* and *Installation & Configuration* manuals, and the **dinit** utility documentation for methods of initializing just those portions of your disk. If only the root block and bitmap are damaged, **chkfsys** will be able to recover the files in most cases. If the root directory or inode file is damaged (the next areas on the disk), recovery may be possible with the **dinit**, **spatch**, and **chkfsys** utilities. Note that such repair requires intimate knowledge of the filesystem structure. Many users would simply recover lost files from a backup at this point. You shouldn't expect to run into such problems; these are very rare events that we have simply tried to anticipate.

Examples:

Check the filesystem on the QNX partition of a hard disk:

```
chkfsys /dev/hd0t77
```

Check the QNX filesystem mounted as the root (**/**) and automatically rebuild the bitmap:

```
chkfsys -rs /
```

Exit status:

0 The filesystem(s) have been checked. An exit status of zero does *not* indicate that no problems were found. It merely indicates that no irrecoverable errors internal to the **chkfsys** utility were encountered.

>0 The filesystem(s) may not have been checked. The **chkfsys** operation may have been interrupted at the request of the user or an internal error (such as running out of memory) may have occurred.

See also:

dcheck, **dinit**, **Fsys**, **spatch**, **zap**

Installation & Configuration

chmod

Change file modes (POSIX)

Syntax:

chmod [-Rv] *mode file*...

Options:

-R	Recursively change file modes. For each *file* that names a directory, **chmod** changes the file mode bits of the directory and all files in the file hierarchy below it.
-v	Verbose. Display to standard output the operations which are being performed.
mode	Represents the change to be made to the file mode of each *file* named (see the Description below).
file	The pathname of a file whose file mode bits are to be modified.

Description:

The **chmod** utility lets you change any or all of the file permission mode bits of one or more files. For each file you name, **chmod** changes the file permission mode bits according to the *mode* operand.

To change a file's permission mode bits, the user of **chmod** must be either the owner of the file or the superuser.

The *mode* option can be either a *symbolic_mode* expression or a non-negative octal integer.

Symbolic Modes

The *symbolic_mode* has the following form:

[[**agou**] **{+-=}** [**rswx**]] [**,***symbolic_mode*]

The options of the symbolic form are:

© 2005, QNX Software Systems

Option	Description
a	user, group, and other access
g	group access
o	other access
u	user access
+	add specified permissions to the group, other, or user category of the specified files
-	remove specified permissions from the group, other, or user category of the specified files
=	set the specified permissions for the group, other, or user category of the specified files
r	read permission
s	set userid or groupid when executed
w	write permission
x	execute permission

The **agou** specification is optional. When it is not supplied, all the permissions (user, group and other) will be affected, but for **+** and **=** operators only those permissions not set in the file creation mask (see **umask**) will be set.

Some examples of symbolic modes:

Make **myfile** executable by all:

```
chmod a+x myfile
```

Remove read permission for group and others:

```
chmod og-r myfile
```

Perform both the above operations, in the order given, on three files: **myfile**, **file2**, and **zzz**:

```
chmod a+x,og-r myfile file2 zzz
```

Octal Modes

Each permission may be specified with an octal number: read = **4**; write = **2**; execute = **1**; no permission = **0**. The octal equivalents are derived by adding the numbers associated with the four basic permissions. The following table illustrates their use:

Octal number	Symbolic	Permission
0	**---**	none
1	**--x**	execute
2	**-w-**	write
3	**-wx**	write/execute
4	**r--**	read
5	**r-x**	read/execute
6	**rw-**	read/write
7	**rwx**	read/write/execute

In the octal form, the three octal digits represent user, group, and other permissions in that order.

Example of octal modes:

Give the user read/write/execute (octal **7** = **rwx**), group read/execute (octal **5** = **r-x**), and other read only (octal **4** = **r--**) for the file **myfile**:

```
chmod 754 myfile
```

© 2005, QNX Software Systems

Setgid and setuid

The following table shows how the setgid and setuid file modes are represented in octal:

Octal number	File mode
2000	setgid
4000	setuid
6000	setgid and setuid

You can combine these file modes with the permission modes described above. For example:

```
chmod 4666 testfile
```

In this case, setuid is set, and the user, group, and other get read/write access.

Exit status:

0 The utility executed successfully and all requested changes were made.

>0 An error occurred.

See also:

chgrp, **chown**, **find**, **ls**, **umask**

Installation & Configuration

chown

Change file ownership (POSIX)

Syntax:

chown [-Rv] *owner*[**:***group*]] *file*...

Deprecated:

chown [-Rv] *owner*[**.***group*]] *file*...

Options:

-R Recursively change ownership of files. For each *file* operand that names a directory, **chown** changes the user ID of that directory and of all files in the file hierarchy below it.

-v Verbose. Display to stdout the operations which are being performed.

owner A username from the user database, or a numeric userid. The **chown** utility changes the owner of each *file* to the user ID of the specified owner.

group A groupname from the user database, or a numeric groupid. The **chown** utility changes the group of each *file* to the groupid of the specified group.

file The pathname of a file whose ownership is to be modified.

Description:

The **chown** utility sets each file's owner and group to the user and groupids specified by the *owner* and *group* operands.

Examples:

Change the owner of file **data** to user 27:

```
chown 27 data
```

Change the owner of the file **data** to **dtdodge**:

```
chown dtdodge data
```

Change the owner of the file **subfile** to **dtdodge** and set the group of the file to **techies**:

```
chown dtdodge:techies subfile
```

Exit status:

0 The utility executed successfully and all requested changes were made.

>0 An error occurred.

Caveats:

If you invoke **chown** with the **-R** option, and **chown** attempts but fails to change the owner or group of a particular file in a specified file hierarchy, it continues to process the remaining files in the hierarchy. The **chown** utility can fail to change the user or group of a file if you don't have appropriate permissions.

You must be **root** to change the file ownership from one owner to another.

Normal users can change the ownership of a file *to themselves*, but they can't give a file away to another user by changing the file ownership.

For compatibility with some other implementations of **chown**, a deprecated syntax allows a period (**.**) to be used instead of a colon (**:**) to separate *user* and *group* (e.g. *user***:***group* and *user***.***group* are both allowed). However, be aware that if a userid contains a period, it may be specified either alone or in conjunction with a group using **:**, but may not be used in conjunction with a group using **.**. For instance if there was a userid **my.name** and a group **tech**, you could do a **chown my.name myfile** or **chown my.name:tech myfile**, but *not* **chown my.name.tech myfile**.

See also:

chgrp, **chmod**, **find**

ci

© 2005, QNX Software Systems

Check in RCS revisions (UNIX)

Syntax:

ci [*options***]** *file***...**

Options:

The **ci** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-d[*date*] Use *date* for the checkin date and time. The date is specified in free format (see "Specifying date and time" in **co**). This option is useful for lying about the checkin date; it's also useful for option **-k** if no date is available. If you omit the *date* argument, **ci** uses the current date and time.

-f[*rev*] Force a deposit. The new revision is deposited even if it's the same as the preceding one.

-i[*rev*] Initial checkin; report an error if the RCS file already exists. This avoids race conditions in certain applications.

-I[*rev*] Run interactively; prompt and question the user even if the standard input isn't a terminal.

-j[*rev*] Just checkin and do not initialize; report an error if the RCS file does not already exist.

-k[*rev*] Search the working file for keyword values to determine the file's revision number, creation date, state, and author, and assign these values to the deposited revision, rather than compute them locally.

-l[*rev*] ("el") Act as if **-r** were specified, except perform an additional **co -l** for the deposited revision. Thus, the deposited revision is immediately checked out again

and locked. This is useful for saving a revision although you want to continue editing it after the checkin.

-m*msg* Use the string *msg* as the log message for all revisions checked in.

-M[*rev*] Set modification date to that of the retrieved revision.

-n *name* Assign the symbolic name *name* to the number of the checked-in revision. If *name* is already assigned to another number, **ci** prints an error message.

-N*name* Act as if **-n** were specified, except override a previous assignment of *name*.

-q[*rev*] Be quiet; don't print diagnostic output.

-r[*rev*] Assign the revision number *rev* to the checked-in revision, release the corresponding lock, and delete the working file. This is the default. The *rev* argument may be symbolic, numeric, or mixed.

-s*state* Set the state of the checked-in revision to *state* (default is **Exp**).

-t[*file*] Write descriptive text from the contents of *file* into the RCS file, deleting the existing text. The filename may *not* begin with a dash (**-**). If you don't specify a file, **ci** prompts you to enter a description.

-T Set the RCS file's modification time to the new revision's time if the former precedes the latter and there is a new revision; preserve the RCS file's modification time otherwise.

If you have locked a revision, **ci** usually updates the RCS file's modification time to the current time, because the lock is stored in the RCS file and removing the lock requires changing the RCS file. This can create an RCS file newer than the working file in one of

two ways: first, **ci -M** can create a working file with a date before the current time; second, when reverting to the previous revision the RCS file can change while the working file remains unchanged. These two cases can cause excessive recompilation caused by a **make** dependency of the working file on the RCS file. The **-T** option inhibits this recompilation by lying about the RCS file's date. Use this option with care; it can suppress recompilation even when a checkin of one working file should affect another working file associated with the same RCS file. For example, suppose the RCS file's time is 01:00, the (changed) working file's time is 02:00, some other copy of the working file has a time of 03:00, and the current time is 04:00. Then **ci -d -T** sets the RCS file's time to 02:00 instead of the usual 04:00; this causes **make** to think (incorrectly) that the other copy is newer than the RCS file.

-u[*rev*] Act as if **-l** were specified, except don't lock the deposited revision. This is useful for reading the working file immediately after checkin.

-V[*n*] Report RCS version; or, if a number is supplied immediately after **-V** (no spaces), emulate RCS version *n*, where *n* may be 3, 4, or 5. This option can be useful when you're interchanging RCS files with other users who are running older versions of RCS (see also "Determining RCS version" in **co**).

-w*login* Use *login* for the author field of the deposited revision. This option is useful for lying about the author; it's also useful for the **-k** option if no author is available.

-x*suffixes* Specifies the suffixes for RCS files. A nonempty suffix matches any pathname ending in the suffix. An empty suffix matches any pathname of the form **RCS/***path* or *path1***/RCS/***path2*. The *-x* option can specify a list of suffixes separated by **/**. For example, **-x,v/** specifies

two suffixes: **,v** and the empty suffix. If two or more suffixes are specified, they are tried in order when looking for an RCS file; the first one that works is used for that file. If no RCS file is found but an RCS file can be created, the suffixes are tried in order to determine the new RCS file's name. The default for *suffixes* is **,v/**.

-z*zone* Specifies the date output format in keyword substitution, and specifies the default time zone for *date* in the **-d***date* option. The *zone* should be empty, a numeric UTC offset in the form

[**+|-**]*hh***:***mm*

or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication.

The **-z** option does not affect dates stored in RCS files, which are always UTC.

file The pathname of an RCS file and/or a working file.

Description:

The **ci** utility stores new revisions into RCS files. Each filename ending in **,v** is taken to be an RCS file. All others are assumed to be working files containing new revisions. The **ci** utility deposits the contents of each working file into the corresponding RCS file. If only a working file is given, **ci** tries to find the corresponding RCS file in an **RCS** subdirectory and then in the working file's directory.

If the RCS file doesn't exist, **ci** creates it and deposits the contents of the working file as the initial revision (default number: 1.1). The access list is initialized to empty. Instead of the log message, **ci** requests descriptive text (see "Descriptive text," below).

For each revision, **ci** prints the RCS file, the working file, and the number of both the deposited and the preceding revision.

File naming

You can specify pairs of RCS files and working files in three ways (see also the Example section of **co**):

1. Both the RCS file and the working file are given. The RCS filename is of the form *path1/workfile*,**v** and the working filename is of the form *path2/workfile* where *path1/* and *path2/* are possibly different or empty paths and *workfile* is a filename.
2. Only the RCS file is given. Then the working file is created in the current directory and its name is derived from the name of the RCS file by removing *path1/* and the **,v** suffix.
3. Only the working file is given. Then **ci** looks for an RCS file of the form *path2*/**RCS**/*workfile*,**v** or *path2/workfile*,**v** (in this order).

If the RCS file is specified without a path in 1 and 2 (above), then **ci** looks for the RCS file first in the directory **./RCS** and then in the current directory.

For ci to work...

For **ci** to work, the caller's login must be on the access list, unless the access list is empty or the caller is the superuser or the owner of the file. To append a new revision to an existing branch, the tip revision on that branch must be locked by the caller. Otherwise, only a new branch can be created. This restriction isn't enforced for the owner of the file if non-strict locking is used (see **rcs**). A lock held by someone else may be broken with the **rcs** command.

Forcing a deposit

Normally, **ci** checks whether the revision to be deposited differs from the preceding one. If they're the same, **ci** reverts to the preceding revision instead of creating a new revision. To force a deposit, you

use the **-f** option. The **ci** utility will deposit a revision that's the same as the preceding revision only if **-f** is specified.

Log messages

For each revision deposited, **ci** prompts you for a log message. The log message you enter should summarize the change and must be terminated by EOF or by a line containing only a single dot (**.**). If several files are checked in, **ci** asks whether to reuse the previous log message. If the standard input isn't a terminal, **ci** suppresses the prompt and uses the same log message for all files. You can also use the **-m** option to specify the same log message for all revisions checked in.

Descriptive text

The **-t** option, in both its forms, is effective only during an initial checkin; it's silently ignored otherwise. During the initial checkin, if **-t** isn't given, **ci** obtains the text from standard input, terminated by EOF or by a line containing only a dot (**.**). The user is prompted for the text if interaction is possible; see **-I**. For backward compatibility with older versions of RCS, a bare **-t** option is ignored.

Keywords

The **-k** option generates a default checkin message noting the login of the caller and the actual checkin date. This option is useful for software distribution. A revision that is sent to several sites should be checked in with the **-k** option at these sites to preserve the original number, date, author, and state. The extracted keyword values and the default log message may be overridden with the options **-d**, **-m**, **-s**, **-w**, and any option that carries a revision number. For more information on keywords, see the **co** utility.

Options using *rev*

You can specify the number of the deposited revisions by any of the options **-f**, **-I**, **-k**, **-l**, **-q**, **-r**, or **-u**. The following rules apply using *rev*:

© 2005, QNX Software Systems

If *rev*:	Then:
is a revision number	it must be higher than the latest one on the branch to which *rev* belongs or must start a new branch.
is a branch rather than a revision number	the new revision is appended to that branch. The level number is obtained by incrementing the tip revision number of that branch.
indicates a nonexisting branch	that branch is created with the initial revision numbered *rev*.1.
is omitted	**ci** tries to derive the new revision number from the caller's last lock. If the caller has locked the tip revision of a branch, the new revision is appended to that branch. The new revision number is obtained by incrementing the tip revision number. If the caller locked a non-tip revision, a new branch is started at that revision by incrementing the highest branch number at that revision. The default initial branch and level numbers are 1.
is omitted and the caller has no lock, but owns the file and locking isn't set to strict	the revision is appended to the default branch (normally the trunk*; see the **-b** option of **rcs**).

* Note that on the trunk, revisions can be appended to the end, but not inserted.

File modes

An RCS file created by **ci** inherits the read and execute permissions from the working file. If the RCS file already exists, **ci** preserves the RCS file's read and execute permissions. The **ci** utility always turns off all write permissions of RCS files.

Files:

Several temporary files may be created. A semaphore file is created in the directory containing the RCS file. The effective user and group must be able to read the RCS file and to search and write the directory containing the RCS file. Normally, the real user and group must be able to read the working file and to search and write the directory containing the working file. The effective user and group are the same as the real user and group unless your RCS executables have setuid or setgid privileges. These privileges yield extra security if RCS files are protected so that only the effective user and group can write RCS directories. Further protection can be achieved by granting access only to the effective user and group.

The **ci** utility never changes an RCS file or a working file; instead, it unlinks the file and creates a new one. This strategy breaks hard links to such files, but doesn't affect symbolic links.

Exit status:

0 Successful completion.

>0 An error occurred.

The exit status is zero only if *all* operations were successful.

Author:

GNU

See also:

co, **ident**, **rcs**, **rcsdiff**, **rcsmerge**, **rlog**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience* 15, 7, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly & Associates, 1995.

Syntax:

cksum [[-9|-12] | -o *algorithm***] [-q | -v] [***file***...]**

Options:

-9	*Deprecated*. Use the algorithm from Draft 9.
-12	("twelve") *Deprecated*. Use the algorithm from Draft 12
-o *algorithm*	Use the specified algorithm.

Algorithm:	Action:
1	Use historic 16-bit checksum algorithm.
2	Use historic 32-bit checksum algorithm.
9	Use 1003.2 draft 9 algorithm (QNX 4.0)
11	Use 1003.2 draft 11 algorithm
12	Use 1003.2 draft 12 algorithm
92	Use 1003.2-1992 standard algorithm (Default)
4.1	Use old QNX **cksum** algorithm (QNX 4.10-4.21)

-q	Quiet. Do not display header (counteracts **-v**). (Default)
-v	Verbose. Display a header which states the algorithm used and names the output columns.
file	The pathname of a file to be checked. If no files are specified, the standard input is used.

All the options to **cksum** are QNX extensions to the POSIX 1003.2 **cksum**.

Description:

The **cksum** utility writes one line to standard output for each file you specify. This line will contain the checksum of the file, as well as the file size and the name of the file being checked. The format of this output varies slightly depending on the command-line options specified to **cksum** as follows:

-o algorithm:	Filesize units:	Output format:
1	k bytes	**%lu %lu %s**
2	512-byte blocks	**%lu %lu %s**
all others	bytes	**%10lu %10lu %s**

If you don't specify any files, **cksum** processes standard input; no filename is given in the output line.

The **cksum** utility lets you quickly compare a suspect version of a file to a trusted version of the same file. You can also use **cksum** to check files after they have been transferred by modem, restored from backup media, or unpacked from a compressed form. The utilities that perform these operations have their own checks, but **cksum** serves as a useful independent checking mechanism.

If you wish to perform a byte-by-byte comparison of files, you can use the **cmp** utility.

Examples:

Report the checksum of the file **archive.tar.F**:

```
cksum archive.tar.F
```

Files:

The data to be used for the checksum calculations are read from standard input if no *file* is specified on the command line. If *file*s are specified, **cksum** will open and read from those files and the standard input will not be used.

The results of the calculations are written to standard output in the format detailed above in the **cksum** description.

Errors encountered during execution will result in diagnostic messages being written to standard error.

Environment variables:

CKSUM default command-line option to use for checksums. This environment variable is supported to remedy those situations where a recent version of the **cksum** utility must be installed in an older system which has an abundance of programs which rely on the pre-QNX 4.2 default output. The use of this environment variable is deprecated, and it may be dropped in a future release.

Exit status:

0 All files were processed successfully.

>0 An error occurred.

See also:

cmp, **diff**, **wc**

clock

© 2005, QNX Software Systems

Display date and time on console (QNX)

Syntax:

clock [-b *color***] [-f** *color***] [-s] [-u] [+***format***]**

Options:

-b *color* — Set background color, where *color* can be **black**, **blue**, **green**, **red**, **cyan**, **magenta**, **brown**, or **white**. The default background color is **black**.

-f *color* — Set foreground color, where *color* can be **black**, **blue**, **green**, **red**, **cyan**, **magenta**, **brown**, **white**, **yellow**, **dgray** (dark gray), **bblue** (bright blue), **bgreen**, **bred**, **bcyan**, **bmagenta**, or **bwhite**. The default foreground color is **white**.

-s — Sleep 1 second between display updates.

-u — Display Coordinated Universal Time (UTC). UTC is the standard term for Greenwich Mean Time (GMT).

+*format* — The format in which the date and time are to be displayed. For an explanation of how to use this option, see the **date** utility.

Description:

The **clock** utility maintains a display of the date and time in the top right corner of your console display. This utility automatically adjusts its priority to level 1 so that it doesn't compete with other processes.

With the exception of the **-b** *color*, **-f** *color* and **-s** options, **clock** works much like the **date** utility-both let you format the date and time with the same **+***format* option. (For information on using this option, see **date**.)

To set the date and time at the system level, use the **date** utility. To transfer the system time to the battery backed-up hardware clock and vice-versa, use the **rtc** utility.

The **clock** utility is a link to **date**.

Since **clock** writes directly to video memory, it may be run only by **root**.

Examples:

Display the clock in bright cyan on blue:

```
clock -f bcyan -b blue &
```

Display the clock in green on black, in 24-hour format:

```
clock -f green "+%X" &
```

Caveats:

Some machines have problems doing DMA (e.g. floppy I/O) while video memory is being accessed. If you are running into problems with this the **-s** option may be used to make video updates less frequent. When **-s** is not used, clock will update the screen once per "tick" (typically 100 times per second). Be aware that other video updates may cause a problem (windows, console writes) and that you are taking a chance with your data if you use such hardware.

See also:

date, **rtc**

cmp

© 2005, QNX Software Systems

Compare two files (POSIX)

Syntax:

cmp [-l|-s] *file1 file2*

Options:

-l ("el") Print the byte position (in decimal) and the differing bytes (octal) for all differences (not just the first one) between the two files.

-s Be silent. Return exit status only.

file1 The pathname of the first file to be compared. If *file1* is the dash character (**-**), the standard input is used.

file2 The pathname of the second file to be compared.

Description:

The **cmp** utility compares two files. If you don't specify any options, **cmp** behaves as follows:

- If the two files are the same, **cmp** writes no output.
- If the files differ, **cmp** writes to standard output the byte and line number at which the first difference occurred. Bytes and lines are numbered, beginning at 1.

If you specify both the **-s** and **-l** options, nothing is printed (no long output).

Examples:

Compare the files **test.dat** and **save.dat**:

```
cmp test.dat save.dat
```

Files:

The standard input will be used only if one of the command-line *file* parameters is **-**.

The results of the file comparison are written to the standard output.

If errors occur, diagnostic messages will be written to the standard error.

Exit status:

0 The files are identical.

1 The files differ. This includes cases where one file is identical to the first part of the other. In such cases, if you haven't specified the **-s** option, **cmp** writes to standard error a message that EOF was reached in the shorter file (before any differences were found).

>1 An error occurred.

See also:

cksum, **diff**, **wc**

co

© 2005, QNX Software Systems

Check out RCS revisions (UNIX)

Syntax:

co [*options***]** *file***...**

Options:

The **co** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-d*date* Retrieve the latest revision on the selected branch whose checkin date/time is less than or equal to *date*. The date and time may be given in free format.

-f[*rev*] Force an overwrite of the working file. This option is useful when used with **-q** (see also "File modes," below).

-I[*rev*] Run interactively; prompt and question the user even if the standard input isn't a terminal.

-j*joinlist* Generate a new revision that is the join of the revisions on *joinlist*. (This option has been largely made obsolete by **rcsmerge**, but is retained for backwards compatibility.)

-kb Generate a binary image of the old keyword string. This acts like **-ko**, except it performs all working file input and output in binary mode. This makes little difference on QNX. Note that under QNX, **rcsmerge** normally refuses to merge files when **-kb** is in effect.

-kk Generate only keyword names in keyword strings; omit their values (see "Keyword substitution," below). For example, for the Revision keyword, generate the string **$Revision$** instead of **$Revision: 5.4 $**. When you're comparing different revisions of a file, this

option is useful for ignoring differences due to keyword substitution.

-kkv Generate keyword strings using the default form (e.g. **$Revision: 5.4 $**) for the Revision keyword. A locker's name is inserted in the value of the **Header**, **Id**, and **Locker** keyword strings only as a file is being locked (i.e. by **ci -l** and **co -l**). This is the default.

-kkvl Act as if **-kkv** were specified, except always insert a locker's name if the given revision is currently locked.

-ko Generate the old keyword string present in the working file just before it was checked in. For example, for the Revision keyword, generate the string **$Revision: 1.1 $** instead of **$Revision: 5.4 $** if that is how the string appeared when the file was checked in.

This option is useful for binary file formats that can't tolerate any changes to substrings that happen to take the form of keyword strings.

-kv Generate only keyword values for keyword strings. For example, for the Revision keyword, generate the string **5.4** instead of **$Revision: 5.4 $**. This can help generate files in programming languages where it's hard to strip keyword delimiters like **$Revision: $** from a string.

Since further keyword substitution can't be performed once the keyword names are removed, you should use this option with care. Because keywords could be lost, this option can't be combined with **-l**. The option also turns off the owner write permission of the working file. To edit the file later, check it out again without **-kv**.

-l[*rev*] ("el") Act as if **-r** were specified, except lock the retrieved revision for the caller.

-M[*rev*] Set the modification time on the new working file to be the date of the retrieved revision. Use this option with care; it can confuse **make**.

-p[*rev*] Print the retrieved revision on the standard output rather than store it in the working file. This option is useful when **co** is part of a pipe.

-q[*rev*] Be quiet; don't print diagnostics.

-r[*rev*] Retrieve the latest revision whose number is less than or equal to *rev*. If *rev* indicates a branch rather than a revision, the latest revision on that branch is retrieved. If *rev* is omitted, the latest revision on the default branch is retrieved (see the **-b** option of **rcs**).

A revision is composed of one or more numeric or symbolic fields separated by periods. The numeric equivalent of a symbolic field is specified with the **-n** option of the **ci** and **rcs** commands.

-s*state* Retrieve the latest revision on the selected branch whose state is set to *state*.

-T Preserve the modification time on the RCS file even if the RCS file changes because a lock is added or removed. This option can suppress extensive recompilation caused by a **make** dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the change of lock would mean a change to keyword strings in the other working file.

-u[*rev*] Act as if **-r** were specified, except *un*lock the retrieved revision if it was locked by the caller. If you omit *rev*, **-u** retrieves the latest revision locked by the caller; if no such lock exists, it retrieves the latest revision on the default branch.

-V[*n*] Report RCS version; or, if a number is supplied immediately after **-V** (no spaces), emulate RCS version *n*, where *n* may be **3**, **4**, or **5**. This option can be useful when you're interchanging RCS files with other users who are running older versions of RCS.

-w [*login*] Retrieve the latest revision on the selected branch that was checked in by the user with login name *login*. If the *login* argument is omitted, the caller's login is assumed.

-x*suffixes* Use *suffixes* to characterize RCS files. See **ci** for details.

-z*zone* Specify the date output format in keyword substitution, and specify the default time zone for *date* in the **-d***date* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication.

The **-z** option does not affect dates stored in RCS files, which are always UTC.

Description:

The **co** utility retrieves a revision from each RCS file and stores it into the corresponding working file. Each filename ending in **,v** is taken to be an RCS file; all other files are assumed to be working files. If only a working file is given, **co** tries to find the corresponding RCS file in the directory **./RCS** and then in the current directory.

The RCS filename, the working filename, and the revision number retrieved are written to the diagnostic output.

File naming

You can specify files in the same way that you specify files for the **ci** utility:

1. Both the RCS file and the working file are given. The RCS filename is of the form *path1/workfile*,**v** and the working filename is of the form *path2/workfile* where *path1/* and *path2/* are possibly different or empty paths and *workfile* is a filename.
2. Only the RCS file is given. Then the working file is created in the current directory and its name is derived from the name of the RCS file by removing *path1/* and the **,v** suffix.
3. Only the working file is given. Then **co** looks for an RCS file of the form *path2*/**RCS**/*workfile*,**v** or *path2/workfile*,**v** (in this order).

As in **ci**, if the RCS file is specified without a path in 1. and 2., then **co** looks for the RCS file first in the directory **./RCS** and then in the current directory (this also applies to **ci**).

File locking

Revisions of an RCS file may be checked out locked or unlocked. Locking a revision prevents overlapping updates. A revision checked out for reading or processing (e.g. compiling) need not be locked. A revision checked out for editing and later checkin must be locked. Checkout with locking fails if the revision to be checked out is currently locked by another user (a lock may be broken with **rcs**). Checkout with locking also requires the caller to be on the access list of the RCS file, *unless* either of the following is true:

- the caller is the owner of the file or the superuser
- the access list is empty

Checkout without locking isn't subject to access list restrictions and isn't affected by the presence of locks.

Selecting revisions

A revision is selected by options for revision or branch number, checkin date/time, author, or state. When you combine several selection options, **co** retrieves the latest revision that satisfies all of them. If no selection option is specified, **co** retrieves the latest revision on the default branch (normally the trunk, see the **-b** option of **rcs**). You can attach a revision or branch number to any of the options **-f**, **-I**, **-l**, **-p**, **-q**, **-r**, or **-u**. The options **-d** (date), **-s** (state), and **-w** (author) retrieve from either of the following:

- a single branch, the selected branch, which is specified by one of **-f**, ..., **-u**
- the default branch

A **co** command applied to an RCS file with no revisions creates a zero-length working file.

Determining RCS version

To see which version of RCS your correspondents are running, have them invoke **rlog** on an RCS file. The following table shows you what to look for:

If:	Version is:
None of the first few lines of output contain the string branch	3
The dates' years have just two digits	4
Neither of the above is true	5

Also note the following:

- An RCS file generated while emulating version 3 will lose its default branch.

- An RCS revision generated while emulating version 4 or earlier will have a time stamp that is off by up to 13 hours.
- A revision extracted while emulating version 4 or earlier will contain dates of the form *yy*/*mm*/*dd* instead of *yyyy*/*mm*/*dd* and may also contain white space in the substitution for the **Log** keyword.

Specifying date and time

When you specify date and time through option **-d**, remember that time zone LT stands for local time. Note also that other common time zone names are understood. For example, the following dates are equivalent if local time is January 11, 1997, 8 pm Pacific Standard Time (eight hours west of UTC):

```
8:00 pm LT
4:00 AM, Jan. 12, 1997
1997/01/12 04:00:00
Thu Jan 11 20:00:00 1997 LT
Thu Jan 11 20:00:00 PST 1997
Fri Jan 12 04:00:00 UTC 1997
Thu, 11 Jan 1997 20:00:00 -0800
Fri-JST, 1997, 1pm Jan 12
12-January-1997, 04:00-WET
```

Most fields in the date and time may be defaulted. The default time zone is UTC. The other defaults are determined in the order year, month, day, hour, minute, and second (most- to least-significant). You must provide at least one of these fields. For omitted fields that are of higher significance than the highest provided field, the time zone's current values are assumed. For all other omitted fields, the lowest possible values are assumed. For example, the date:

```
20, 10:30
```

defaults to 10:30:00 UTC of the 20th of the UTC time zone's current month and year. You must quote the date/time if there are spaces.

Keyword substitution

The **co** utility always performs keyword substitution. Strings of the form **$***keyword***$** and **$***keyword***:**...**$** embedded in the text are replaced with strings of the form **$***keyword***:***value***$** listed below. Note that you can embed keywords in literal strings or comments to identify a revision.

Initially, you enter strings of the form **$***keyword***$**. On checkout, **co** replaces these strings with strings of the form **$***keyword***:***value***$**. If a revision containing strings of the latter form is checked back in, the value fields will be replaced during the next checkout. Thus, the keyword values are automatically updated on checkout. This automatic substitution can be modified by the **-k** options.

The following table lists the keywords and their corresponding values:

Keyword	Corresponding value
$Author$	The login name of the user who checked in the revision.
$Date$	The date and time (UTC) the revision was checked in.
$Header$	A standard header containing the full pathname of the RCS file, the revision number, the date (UTC), the author, the state, and the locker (if locked).
Id	(same as **$Header$**, except that the RCS filename is without a path)
$Locker$	The login name of the user who locked the revision (empty if not locked).

continued...

Keyword	Corresponding value
Log	The log message supplied during checkin, preceded by a header containing the RCS filename, the revision number, the author, and the date (UTC). Existing log messages aren't replaced. Instead, the new log message is inserted after **$Log: ... $**. This is useful for accumulating a complete change log in a source file.
$RCSfile$	The name of the RCS file without a path.
$Revision$	The revision number assigned to the revision.
$Source$	The full pathname of the RCS file.
$State$	The state assigned to the revision with the **-s** option of **rcs** or **ci**.

Join lists

You can use the **-j***joinlist* option to generate a revision that is the join of several other revisions. The *joinlist* is a list of pairs, separated by commas; each pair is of the form:

rev2:*rev3*

where *rev2* and *rev3* are (symbolic or numeric) revision numbers. For the initial such pair, *rev1* denotes the revision selected by options **-f**, ..., **-w**. For all other pairs, *rev1* denotes the revision generated by the previous pair. (Thus, the output of one join becomes the input of the next.)

For each pair, **co** joins revisions *rev1* and *rev3* with respect to *rev2*. This means that all changes that transform *rev2* into *rev1* are applied to a copy of *rev3*. This is particularly useful if *rev1* and *rev3* are the ends of two branches that have *rev2* as a common ancestor. If *rev1*<*rev2*<*rev3* are on the same branch, joining generates a new revision that's like *rev3*, but with all changes that lead from *rev1* to *rev2* undone. If changes from *rev2* to *rev1* overlap with changes from

rev2 to *rev3*, **co** prints a warning and includes the overlapping sections, delimited by these lines:

<<<<<<< *rev1*
=======
>>>>>>> *rev3*

For the initial pair, *rev2* may be omitted. The default is the common ancestor. If any of the arguments indicate branches, the latest revisions on those branches are assumed. The options **-l** and **-u** lock or unlock *rev1*.

File modes

The working file inherits the read and execute permissions from the RCS file. In addition, the owner write permission is turned on, unless either of the following is true:

- **-kv** is set
- the file is checked out unlocked and locking is set to strict (see **rcs**)

If a file with the name of the working file already exists and has write permission, **co** asks if it can delete the existing writable file. If the existing working file isn't writable or **-f** is given, the working file is deleted without asking.

Examples:

Let's say the current directory contains a subdirectory **RCS** with an RCS file **io.c,v**. All of the following commands would retrieve the latest revision from **RCS/io.c,v** and store it into **io.c**:

```
co io.c

co RCS/io.c,v

co io.c,v
```

```
co io.c RCS/io.c,v

co io.c io.c,v

co RCS/io.c,v io.c

co io.c,v io.c
```

Files:

The **co** utility accesses files much as **ci** does, except that it doesn't need to read the working file.

Exit status:

0 Successful completion.

>0 An error occurred.

The exit status is zero only if *all* operations were successful.

Author:

GNU

Caveats:

The **-d** option sometimes gets confused. In addition, it accepts no date before 1970.

Links to the RCS files and working files aren't preserved.

There's no way to selectively suppress the expansion of keywords, except by writing them differently. This is done by embedding another character into the keyword.

See also:

ci, **date**, **ident**, **rcs**, **rcsdiff**, **rcsmerge**, **rlog**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience* 15, 7, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly & Associates, 1995.

comm

© 2005, QNX Software Systems

Select or reject lines common to two files (POSIX)

Syntax:

comm [-1] [-2] [-3] *file1 file2*

Options:

-1 (the number one) Suppress writing of lines found only in *file1*.

-2 Suppress writing of lines found only in *file2*.

-3 Suppress writing of lines found in both *file1* & *file2*.

file1 A pathname of the first file to be compared. If *file1* is **-**, the standard input is used.

file2 A pathname of the second file to be compared. If *file2* is **-**, the standard input is used.

Description:

The **comm** utility reads *file1* and *file2*, which must be ordered in collating sequence (see the **sort** utility), and produces three text columns as output.

This column:	Contains:
1	lines found only in *file1*
2	lines found only in *file2*
3	lines found in both files

Examples:

Print only the lines common to both files:

```
comm -12 test.dat save.dat
```

Files:

The standard input is used only if specified as a dash (**-**) command-line *file* parameter.

All input files must be text files for **comm** to produce a meaningful result.

The results of the comparison will be written to standard output.

Errors will result in diagnostic messages being written to standard error.

Exit status:

0 All input files were successfully output as specified.

>0 An error occurred.

See also:

cmp, **diff**, **sort**, **uniq**

cp

© 2005, QNX Software Systems

Copy files (POSIX)

Syntax:

cp [-f|-i] [-Rrp] [*QNX extensions***]** *source_file target_file*

cp [-f|-i] [-Rrp] [*QNX extensions***]** *source_file***...** *target_dir*

Options:

-a *date/time*
(QNX extension — *Deprecated*. Use **find | pax -rw**.) Copy only those files that were last modified *after* the specified *date/time*, where *date/time* is:

[**UTC**] *dd*/*mm*/*yy* [*HH*:*MM*:*SS*]

specified in 24 hour time (the time portion is optional). To use Coordinated Universal Time, specify **UTC**; to use the local time, don't include **UTC**.

If you're running **cp** from the shell, and you want to specify both the date and the time, you must use the quotation marks. If you want to specify only the date (*dd/mm/yy*), quotation marks aren't needed.

-A
(QNX extension) Preserve source file access times, if possible. Cannot be used with **-O**. The access and modification times of source files and directories will be memorized then restored when **cp** has finished reading them.

-b *date/time*
(QNX extension — *Deprecated*. Use **find | pax -rw**.) Copy only those files that were last modified *before* the specified *date/time* (see the **-a** *date/time* option).

-B
(QNX extension — *Deprecated*.) Use a small copy buffer (2K) instead of grabbing a large buffer (16K). If you don't specify this option and there

isn't enough free memory to use a 16K buffer, **cp** dynamically reduces its buffer by 1K increments until there's enough memory available (down to a minimum of 1K).

When even using **-B** fails, one can resort to using the **cat** utility to copy data.

-c (QNX extension) Create any directories necessary to open the destination path. For example, if the directory **/home/eric** doesn't exist, and you enter:

```
cp -c file /home/eric/source/file
```

cp performs the equivalent of:

```
mkdir -p /home/eric/source
cp file /home/eric/source/file
```

-d (QNX extension — *Deprecated*.) Distribute source files to multiple destinations. Ordinarily, you can specify only one destination file or directory to **cp**. But when you specify **-d**, and *only* when you specify **-d**, **cp** treats the word "**to**" as a special marker separating source files and source directories from destinations. For example:

```
cp -d myprog to //61/hd/bin //73/hd/bin //15/hd/bin
```

performs the equivalent of:

```
cp myprog //61/hd/bin
cp myprog //73/hd/bin
cp myprog //15/hd/bin
```

When you don't specify **-d**, the word "**to**" is treated like any other filename.

© 2005, QNX Software Systems

-D (QNX extension) This option, when used in conjunction with **-R**, controls whether **cp** will or will not descend past device boundaries while traversing a directory tree. When the **POSIX_STRICT** environment variable is set, **cp** will ignore device boundaries (i.e. will copy everything). When **POSIX_STRICT** isn't set, **cp** will by default *not* descend past device boundaries.

Specifying **-D** will cause **cp** to descend past device boundaries. For example:

```
cp -R //1/ //2/backup
```

will, when **//1/** is a local hard disk, cause the contents of the disk to be backed up, but the **//1/dev** directory will be skipped, since it does not exist on the hard disk device. The addition of **-D**, as follows:

```
cp -RD //1/ //2/backup
```

would cause **cp** to also attempt to back up the contents of the **//1/dev** directory. Note that in this particular example only the disk devices (block special files) would actually have their data backed up to files in **//2/backup/dev** because **cp** won't copy character special files on recursive copies.

-f Force the unlinking of the destination file prior to copying. This option prevents interactive prompting (unless **-i** is also specified) but does *not* disable diagnostic messages. Note that this behavior follows an early POSIX draft and does not comply with POSIX 1003.2-1992. The standard specifies that the file will be unlinked *only* if a file descriptor for the destination file cannot be obtained (i.e. **cp** cannot open the destination file for write).

-g *group*[,*group*]

(QNX extension — *Deprecated*. Use **find | pax -rw**.) Copy only files belonging to these groups (where *group* is a groupid or a group name).

-i

Run interactively; always prompt the invoking user for confirmation when the destination path exists, regardless of whether the user has write permission for the destination file. The **-i** option is useful when you want to avoid accidentally clobbering files when copying. When you don't have write permission for the destination file and you answer yes to the prompt, the destination file will be unlinked first. Otherwise, the destination is simply overwritten and truncated.

The combination of **-i** and **-f** works as if only **-i** were specified, except that when you answer **yes** to the prompt, the destination is always unlinked first-even if you have write permission for the destination file. When only **-i** is specified, the destination is unlinked only when you do not have write permission for the destination file. See note about POSIX compatibility of the current **-f** behavior.

-l *n*

("el") (QNX extension — *Deprecated*. Use **find | pax -rw**.) If *source_file* is a directory, and if option -*R* is specified, copy only *n* levels of the directory tree. If you specify **-l 0**, **-R** is defeated; only files at the current level (files named directly on the command line) are copied.

-L

(QNX extension) Attempt to preserve hard links. When **cp** encounters a file that has a link count **>**1, it remembers that file's device ID and serial number (inode). If during the **cp** another file with a link count **>**1 is found matching the serial

number and device ID, **cp** creates a link instead of making a second copy of the file. When the destination media is changed, **cp** wipes its memory of links encountered to that point. (This is significant when making floppy backups, or backups to removable hard disks.)

-m [!]**mode**[,[!]*mode*]...

(QNX extension — *Deprecated*. Use **find | pax -rw**.) Copy only files with these modes (where *mode* is expressed as an octal number or symbolically similar to the format used in the **chmod** utility). If you specify a *mode* with a preceding ! the **cp** utility will not copy any file that has that mode. The symbolic mode is of the form

[[**agou**]=[**rswx**]][,*symbolic_mode*]

as used by the **chmod** utility.

For example, an option of **-m ug=rx** copies only files for which user and group have read and execute permission.

-M qnx|unix (QNX extension) Recursive copies to be done in UNIX or old-style QNX mode. (Default: **qnx**). QNX has, in the past, copied the *contents* of the directories named on the command line into the target directory. UNIX copies the directory itself into the target directory (like **mv**). In either case if there is only one directory being copied and the destination names a directory that doesn't exist, **cp** will create the destination directory and then copy the *contents* of the source directory into the destination directory. At some point in the future a version of QNX will be released which makes **unix** the default for this option.

-n (QNX extension) Copy only if the source is newer than the destination (has a more recent file

modified time), or if the destination doesn't already exist.

-O (QNX extension) Optimize the copy operation; run the copy on the CPU of the destination node or on the node that will most likely speed up the operation. The **cp** utility will not spawn itself on more than one other node, so when multiple destinations (on different nodes) are specified, it tries to make a "best guess" decision as to which node to run on.

-p After copying, attempt to duplicate the modification time and file mode of each input file in the corresponding output file. Also duplicate the ownership of each file if the process is run with the privileges of the superuser (**root**). If the process doesn't have the appropriate privileges, the duplication will fail.

-P [!]*pat*[,[!]*pat*]...

(QNX extension; *Deprecated*. Use **find | pax -rw**) Copy only files matching these patterns (where *pat* is a filename matching pattern as per the *fnmatch()* function; for details, see the section on "Pattern-matching special characters"). If you specify a pattern with a preceding **!**, and if a file you've specified matches that pattern, **cp** doesn't copy the file.

-r Recursively copy directories. If a source file is a special file (e.g. FIFO, named special file), **cp** doesn't create a special file as the destination. Read the section on Recursive Copying and look at the **-M** and **-r** options.

-R If the *source_file* is a directory, recursively copy the directory with the files and subdirectories under it, attempting to preserve special files. QNX doesn't

allow block special files and character special files to be created in this manner. Read the section on Recursive Copying, and see the **-M** and **-r** options.

-s (QNX extension) Run safely; copy only if the existing destination file has write permission. If the file doesn't have write permission, skip the file without prompting.

-t (QNX extension) Don't attempt to duplicate file time and mode if the **-p** option was specified or if the **POSIX_STRICT** environment variable is set to on.

-u *owner*[,*owner*]...

(QNX extension — *Deprecated*. Use **find | pax -rw**.) Copy only files belonging to these owners, where *owner* is a userid or a username.

-v (QNX extension) Be verbose. Display a line of explanatory text every time a file is copied or a directory is created.

-V (QNX extension) Be extra verbose. In addition to doing everything **-v** does, this option displays a running progress counter (% complete) and it also displays lines when **cp** skips a file or a directory (i.e. you can see what **cp** isn't doing as well as what it is doing). For example, if you select options **-R** and **-n**, you'll find that **cp -VRn** is more useful than **cp -vRn** since option **-v** in this case might let **cp** go away and put you back at the prompt without providing you with any feedback.

-x (QNX extension) Copy only if the destination file already exists.

-X (QNX extension) Copy only if the destination file doesn't exist.

source_file	The pathname of a file to be copied. If you want *source_file* to name a directory, you must also specify the **-R** option.
target_file	The pathname to which a single file is copied.
target_dir	The pathname of an *existing* directory that is to contain the output file(s).

The QNX extended options listed above that are deprecated may not be present in future versions of QNX.

Description:

First syntax form:

cp [**-f|-i**] [**-Rrp**] [*QNX extensions*] *source_file target_file*

The **cp** utility copies the contents of the source file to the destination file named by *target_file*. This first syntax form is assumed when the destination file isn't an existing directory and there's only one source file.

Second syntax form:

cp [**-f|-i**] [**-Rrp**] [*QNX extensions*] *source_file... target_dir*

For each *source_file*, **cp** copies the contents of the file to a destination file in the existing directory named by *target_dir*. The destination's filename under the target directory will be the same as its basename (final path component), unless it is a directory (see Recursive Copying). For example:

```
cp dir/dir/myfile /existingdir
```

copies the contents of **dir/dir/myfile** to the file **/existingdir/myfile**.

This second form is assumed when the destination file is an existing directory or when more than one source file is specified.

General

By default cp duplicates file contents and also duplicates the file times of the original file. This is not standard UNIX behavior for **cp**. To disable this, either disable the duplication of file times by using the **-t** or set **POSIX_STRICT** - see Environment variables for details.

Unless the **-R** (recursive) option is specified, **cp** refuses to copy any *source_file* that is a directory.

For duplicating lists of files, see the **pax -rw** utility, which is another POSIX utility for duplicating files. Sets of files which match complex criteria may be selected with the **find** utility and piped into **pax**. The abilities built into **cp** to accomplish the same thing (e.g. pattern matching) are *deprecated* and will be removed in a future release.

What **cp** does when a destination file already exists depends on the options used. When neither **-f** nor **-i** is specified, you'll be prompted only if you don't have write permission for the existing destination file. When this happens, you'll be asked if you want to unlink the file first. If you don't, **cp** goes on to any remaining files. You'll be prompted only if *stdin* is a tty. Otherwise, **cp** will print a diagnostic message to *stderr* and skip that file.

If you're copying to removable media, such as a floppy or removable disk, and the media becomes full, **cp** performs the following steps:

1. Closes and removes the incompletely copied destination file.
2. Prompts you to insert a new diskette. If you type **y** (yes), **cp** will create the necessary directory tree on the new disk and resume copying, starting with the file that was being copied when space ran out. If you type **n**, **cp** will stop copying and exit.

Recursive copying

When doing a recursive copy of a directory, the destination must be a directory. If more than one item is being copied, the directory must already exist. If a single directory is being copied, the **cp** command

will create the destination directory (all intermediate directories must already exist unless the **-c** option is specified).

There are two recursive copying modes available with **cp**. The default is the historical QNX behavior which is to copy the files and directories underneath the source directory to the destination directory. The source directory itself will not be duplicated within the destination directory.

The other mode, specified by the **-Munix** option, causes cp to duplicate the source directory within the destination directory (unless a single directory is being copied and the destination directory does not yet exist, in which case **-Munix** and **-Mqnx** modes do the same thing).

In **-Mqnx** mode, **cp -r /bin //2/bin** will copy the contents of **/bin** to **//2/bin** (so, for example, **/bin/sh** will be copied to **//2/bin/sh** etc.).

In **-Munix** mode, **cp -Munix -r /bin //2/bin** would duplicate **/bin** in **//2/bin**, i.e. the destination would be **//2/bin/bin** and, for example, the file **/bin/sh** would be copied to **//2/bin/bin/sh**. This is analogous to the way the **mv** utility treats destinations.

The default behavior is currently **-Mqnx**, but this will change in a future major revision of QNX.

Pattern-matching special characters

A pattern-matching special character that is quoted is a pattern that matches the special character itself. When *not* quoted, such special characters have special meaning in the specification of patterns. The pattern-matching special characters and the contexts in which they have their special meaning are as follows:

? matches any printable or non-printable character except **newline**

***** matches any string, including the null string

[*bracket_expr***]**

a pattern that matches a single character, except that:

- the exclamation mark (**!**) replaces the circumflex (**^**) in its role as a nonmatching list in the regular expression notation
- the backslash is used as an escape character within bracket expressions

Note that the characters **?**, *****, and **[** are *not* special when used inside a bracket expression.

The concatenation of patterns matching a single character is a valid pattern that matches the concatenation of the characters matched by each of the concatenated patterns. For example, the pattern **a[bc]** matches the strings **ab** and **ac**.

The concatenation of one or more patterns matching a single character with one or more asterisks (*****) is a valid pattern. In such patterns, each asterisk matches a string of zero or more characters, up to the first character that matches the character following the asterisk in the pattern. For example, the pattern **a*d** matches the strings **ad**, **abd**, and **abcd**, but *not* the string **abc**.

When an asterisk is the first or last character in a pattern, it matches zero or more characters that precede or follow the characters matched by the remainder of the pattern. For example, the pattern **a*d*** matches the strings **ad**, **abcd**, **abcdef**, **aaaad**, and **adddd**. The pattern ***a*d** matches the strings **ad**, **abcd**, **efabcd**, **aaaad**, and **adddd**.

Examples:

Copy **file1**, **file2**, and **file3** from the current working directory to the **/home/eric** directory:

```
cp file1 file2 file3 /home/eric
```

Do a raw block copy from the floppy drive on node 4 to the floppy drive on node 6:

```
cp //4/dev/fd0 //6/dev/fd0
```

Perform a backup of the entire contents of the **home** directory to floppy disks (assuming that **/f0** is a mount point for **/dev/f0**), in the (default) QNX recursive-copy mode:

```
cp -rvp /home /f0/home
```

Do the same in UNIX recursive-copy mode:

```
cp -Munix -rvp /home /f0
```

Recursively copy the **/home/eric** directory to the **/home/ejohnson** directory (assuming **/home/ejohnson** does not yet exist) (will work in either **-Munix** or **-Mqnx** mode):

```
cp -rv /home/eric /home/ejohnson
```

Do the same in **-Mqnx** mode if the directory **ejohnson** already exists:

```
cp -Mqnx -rv /home/eric /home/ejohnson
```

Do the same in **-Munix** mode if the directory **ejohnson** already exists:

```
cp -Munix -rv /home/eric/. /home/ejohnson
```

Recursively copy the contents of the current directory into **//2/** in **-Mqnx** *or* **-Munix** modes:

```
cp -Rpv . //2/
```

Do the same in **-Munix** mode only:

```
cp -Munix -Rpv * //2/
```

In the previous example, doing **cp -Mqnx -Rpv * //2/** would have resulted in copying the *contents* of the directories named on the command line into **//2/** (i.e. the file **./bin/ls** would have been copied to **//2/ls**, and the directory **./usr/bin** would have been **//2/bin** in the destination).

Recursively copy the **/home/eric** directory to the **/backups/eric** directory:

```
cp -rv /home/eric /backups/eric
```

Do the same in UNIX-style recursive copy mode:

```
cp -Munix -rv /home/eric /backups
```

Files:

Input files

If you don't specify the **-r** option, and only one source file is named, that source file may be of any filetype.

If you specify the **-r** option, or there's more than one source file, the input files specified by each *source_file* operand, including those files contained within named directories, must be either regular files, block special files, or directories.

If you use the **-R** option, FIFOs will be duplicated in the destination directory structure, but contents of the source FIFOs won't be copied. Any block special or character special files encountered in the source files will result in an error, since **cp** cannot create these at the destination.

Output files

Each newly created output file is one of the following:

- A directory that contains copies of the files and subdirectories — if any — found in the input directory.
- A regular file that has the same contents as the corresponding input file.
- A FIFO that was created because the corresponding input file was a FIFO and **-R** was specified. Data will *not* be copied from the original FIFO to the new FIFO.
- A symbolic link that was created because the corresponding input file was a symbolic link and **-R** was specified. The new symbolic link will reference the same pathname as the original symbolic link.

If an existing destination names some other type of file, **cp** opens it for writing and attempts to copy the contents of the corresponding input file to it.

Environment variables:

POSIX_STRICT

Affects whether file modification times are copied, and, if set to on, causes the QNX Extension options to be disabled.

The setting of the **POSIX_STRICT** environment variable affects the **-p** and **-t** options, as follows:

POSIX_STRICT	Option	Action
set	neither **-p** nor **-t**	if destination doesn't exist, duplicate mode only
set	**-p**	duplicate time, mode; if run by **root**, also duplicate userid, groupid

continued...

POSIX_STRICT	Option	Action
set	**-pt**	if run by **root**, duplicate userid, groupid
set	**-t**	if destination doesn't exist, duplicate mode only
unset	neither **-p** nor **-t**	duplicate time and mode
unset	**-p**	duplicate time, mode; if run by **root**, also duplicate userid, groupid
unset	**-pt**	if run by **root**, duplicate userid, groupid
unset	**-t**	if destination doesn't exist, duplicate mode only

Exit status:

0 All input files were copied successfully.

>0 An error occurred.

Caveats:

If **cp** is copying multiple files or doing a recursive copy, but the **-R** option is *not* specified, **cp** will refuse to copy FIFO and character special files.

If you specify the **-R** option, and **cp** attempts but fails to copy a particular file in a specified file hierarchy, it continues to process the remaining files in the hierarchy.

See also:

find, **mv**, **ln**, **pax**

Syntax:

Read/list an archive:

cpio -i[Bcdfmrtuv] [*pattern***...]**

Write an archive:

cpio -o[Bacv]

Copy files:

cpio -p[adlmuv] *directory*

Options:

-a Reset access times of input files after they've been copied. When the **-l** option is also specified, the access times of linked files aren't reset. Can be used only with the **-o** or **-i** options.

-B Cause input/output to be blocked 5120 bytes to the record. Can be used only with the **-o** or **-i** options for data directed to or from character special files.

-c Write header information in ASCII (Default; option present for compatibility)

-d Create directories as needed. Can be used only with the **-i** or **-p** options.

-f Copy in all files except those in patterns. Can be used only with the **-i** option.

-i Copy in. (Extract files from an archive being read from standard input.)

-l ("el") Whenever possible, link files rather than copy them. Can be used only with the **-p** option.

-m	Retain previous modification times. This option won't work on directories that are being copied. Can be used only with the **-i** or **-p** options.
-o	Copy out. (Write an archive to standard output.)
-p	Pass. Conditionally copy files from a list read from standard input to the destination directory named as an argument to **cpio**.
-r	Interactively rename files. A new name for each file will be requested from the user. Read and write permissions for the controlling terminal (**/dev/tty**) are required for this option. If you type a null line, the file is skipped. Should be used only with the **-i** or **-o** options.
-t	Print a table of contents of the input. No files are created. Can be used only with the **-i** option.
-u	Copy files unconditionally. Usually an older file will not replace a new file with the same name. Can be used only with the **-i** or **-p** options.
-v	Be verbose. Print the names of the affected files. Can be used only with the **-i** option. Provides a detailed listing when used with the **-t** option.
pattern	Simple regular expression given in the name-generating notation of the shell.
directory	The destination directory.

Description:

The **cpio** utility produces and reads files in the format specified by the POSIX **cpio** Archive/Interchange File Format. It operates in three modes.

The **-i** mode (copy in) extracts files from the standard input, which is assumed to be the product of a previous **cpio -o**. Only files with names that match patterns are selected. Multiple patterns may be

specified. If no patterns are specified, the default for patterns is to select all files. The extracted files are conditionally created and copied into the current directory, and possibly any levels below, based on the options used. The permissions of the files will be those stored by the previous **cpio -o** invocation. The owner and group of the files will be that of the current user unless the user has appropriate privileges, which causes **cpio** to retain the owner and group of the files stored by the previous **cpio -o** invocation.

The **-o** mode writes the archive to the standard output.

The **-p** mode (pass) reads the standard input to obtain a list of pathnames of files that are conditionally created and copied into the destination directory based upon the options used.

If an error is detected, the cause is reported and the **cpio** utility will continue to copy other files. The utility will skip over any unrecognized files encountered in the archive.

The following restrictions apply to the **cpio** utility:

- Pathnames are restricted to 256 characters.
- Appropriate privileges are required to copy special files.
- Blocks are reported in 512-byte quantities.
- Leading slashes (/) will be stripped when files are extracted from an archive.

Examples:

Copy out the files listed by the **ls** utility and redirect them to the file **archive**:

```
ls | cpio -o >archive
```

Use the output file **archive** from the **cpio -o** utility, extract those files that match the patterns **memo/al** and **memo/b***, create the directories below the current directory, and place the files in the appropriate directories:

```
cpio -id "memo/al" "memo/b*" <archive
```

Take the filenames piped to **cpio** from the **find** utility and copy or link those files to another directory named **newdir**, while retaining the modification time:

```
find . -depth -print | cpio -pdlmv newdir
```

Exit status:

0 All input files were copied.

2 The utility encountered errors in copying or accessing files or directories. An error will be reported for nonexistent files or directories or for permissions that don't allow the user to access the source or target files.

Caveats:

When **cpio** restores a directory, it matches the permissions of the directory created to those of the original. If that directory lacks write permission, any attempt to copy additional files under that directory will fail. To get around this, save the files under a directory first before the directory itself. If **find** is used to generate pathnames for **cpio**, the **-depth** option should be supplied to **find**.

Note also that the controlling terminal (**/dev/tty**) is used to prompt the user for information when the **-i** or **-r** options are specified.

See also:

cp, **find**, **lockfd**, **pax**, **tar**

Syntax:

cron [-d *crondir***] [-vL] &**

Options:

-d *crondir*	Specifies that **cron** use the named directory instead of **/usr/spool/cron**.
-v	Turn on verbose mode. Log and diagnostic messages will be written to standard error as **cron** operates.
-L	Local. Automatically adds the suffix *.node_ID* to the cron directory name (**/usr/spool/cron** or the filename given to the **-d** option). e.g. If **cron** is being run on node 95 and **-d** is not specified, the cron directory name would be **/usr/spool/cron.95**. This allows cron to be run on multiple machines which share the same filesystem.

Description:

The **cron** server schedules commands to be run at specified times, without user intervention. This server supports user-specific **cron** entries, and runs continuously. The server must be run in the background.

The **cron** server assumes it has sole use of the **/usr/spool/cron** directory. Therefore, only one **cron** server can be run per filesystem containing that directory. The **cron** server is typically run on the network server.

Commands are specified by instructions found in crontab files, which are accessed via the **crontab** utility.

To minimize overhead, **cron** examines the contents of the files in **/usr/spool/cron/crontabs** when it first comes up, and then re-examines only those that have been changed via the **crontab** utility.

Files:

Errors will cause diagnostic messages to be written to standard error. If **-v** is specified, log messages will be written to the standard error. **cron** makes use of data read from the following:

/usr/spool/cron
/usr/spool/cron.*nid*

Each **cron** command assumes it has exclusive use of this directory. The **cron.***nid* variant is used when the **-L** option has been specified.

/usr/spool/cron/cron.allow
/usr/spool/cron.*nid***/cron.allow**

If present, this file will list the only users authorized to have their crontab run. By default, all users are authorized. The **cron.deny** list (below) overrides the setting of the **cron.allow** list.

/usr/spool/cron/cron.deny
/usr/spool/cron.*nid***/cron.deny**

If present, this file will list users who are not authorized to have their crontab run. This list overrides the list of users authorized (the **cron.allow** file).

/usr/spool/cron/crontabs/*

The periodic commands to be run are read out of files found in this directory.

Exit status:

The **cron** utility normally will run indefinitely. However, it will terminate early if errors are encountered in startup, errors are encountered in reading the **crontabs** files, or if it is terminated by a signal.

0 **cron** was successfully and cleanly terminated by a SIGTERM or SIGPWR signal.

>0 An error occurred. A diagnostic message will have been written to standard error.

See also:

crontab

crontab

© 2005, QNX Software Systems

Schedule periodic background work (POSIX)

Syntax:

crontab [-L | -n *nid***] [-d** *cron_dir***] [-u** *user***] [***file***]**

crontab [-L | -n *nid***] [-d** *cron_dir***] [-e | -l | -r] [-u** *user***]**

Options:

-e Edit a user's **crontab** entry. If **-u** is not specified, the invoking user's **crontab** entry will be edited. The **vi** editor will be used unless another editor is named in the **EDITOR** environment variable.

-l ("el") List the crontab entry of the user. Without the **-u** option, the invoking user's entry is listed.

-L (QNX extension) Local. Automatically append the local node ID to the **crontab** directory as the suffix **.***nid*.

-n *nid* (QNX extension) Similar to **-L**, except the node ID specified is used instead of the local node ID.

-r Remove a user's **crontab** entry. Without the **-u** option, the invoking user's entry is removed.

-u *user* Specify the user whose crontab will be acted upon. When submitting a crontab, the new crontab entry replaces or creates that user's crontab. When removing (**-r**) or listing (**-l**) existing crontabs, this specifies which user's crontab to remove or list. Only **root** may use this option.

file The pathname of a file that contains specifications for **crontab** entries (see the Description section for the format for these specifications). If no file is specified, **crontab** uses the standard input.

Description:

The **crontab** utility creates or replaces a user's crontab entry. You can input the new crontab entry by specifying a file that contains specifications for crontab entries. If you don't specify a file, the standard input is used.

Users may use **crontab** if their names appear in the **/usr/spool/cron/cron.allow** file. If that file doesn't exist, the file **/usr/spool/cron/cron.deny** is checked to determine whether the user should be denied access to **crontab**. If neither file exists, only the superuser is allowed to modify crontab entries. If **cron.allow** doesn't exist and **cron.deny** exists and is empty, then global usage is allowed. These permission files consist of one user name per line.

Each command you specify is executed from your home directory using the shell (**/bin/sh**). The **cron** utility supplies a default environment for the shell, defining **HOME**, **LOGNAME**, **SHELL**(=**/bin/sh**), **PATH** (=**:/bin:/usr/bin**) and **TZ**. Users who want to have their **.profile** executed must explicitly do so in their crontab entry.

A **crontab** entry consists of lines of six fields each. The fields are separated by blanks. The first five are integer patterns that specify the following:

- minute (0-59)
- hour (0-23)
- day of the month (1-31)
- month of the year (1-12)
- day of the week (0-6 with 0=Sunday)

Each of these patterns can be an asterisk (meaning all valid values), an element, a list of elements separated by commas, or a range separated by a hyphen (**-**).

An element is either a number or two numbers separated by a hyphen (meaning an inclusive range). You can specify days by two fields (day of the month and day of the week). If both are specified, both are adhered to.

As an example of specifying the two types of days, the following line:

```
0 0 1,15 * 1
```

runs a command at 00:00h on the first and fifteenth of each month, as well as at 00:00h on every Monday. To specify days with only one field, the other field should be set to *****. For example:

```
0 0 * * 1
```

runs a command only on Mondays.

The sixth field of a line in a crontab entry is a string that is executed by the command interpreter at the specified times. A percent character in this field — unless escaped by a backslash — is translated to a **newline** character.

Only the first line (up to a **%** or end-of-line) of the command field is executed by the command interpreter.

Sample crontab entries

Invoke the **calendar** program each day one minute after midnight:

```
1 0 * * * calendar -
```

Display the current time on the system console every 20 minutes:

```
1,21,41 * * * * (echo -n " "; date; echo) > /dev/con1
```

Clean up the UUCP work directories each weekday at 5:30 am:

```
30 5 * * 1-5 /usr/lib/uucp/uuclean
```

Remove any files under **/tmp** that haven't been modified in more than seven days:

```
0 4 * * * find /tmp -mtime +7 -exec rm -f {} \;
```

By default, the output of your commands is mailed to you. You may wish to redirect the standard output and standard error of your commands to avoid getting the mailed output.

Examples:

List your own crontab entry:

```
crontab -l
```

Files:

When an error occurs a diagnostic message will be written to the standard error.

The standard input may be read for the content of **crontab** files when they are being created (none of **-e**, **-l** or **-r** are specified) and no *file* is specified on the command line.

When a **crontab** file is being edited, the editor invoked by **crontab** will inherit **crontab**'s standard input, standard output and standard error.

The following files relative to one of **/usr/spool/cron**, **/usr/spool/cron.***nid* or the directory named in the **-d** option are used by **crontab**:

cron.allow If this file exists, it is read by **crontab** to determine the exclusive list of userids who are allowed to schedule **cron** jobs.

cron.deny If **cron.allow** doesn't exist, **crontab** reads this file to determine a list of userids who are *NOT* allowed to schedule **cron** jobs.

© 2005, QNX Software Systems

crontabs/*userid*

The **crontab** utility may create, edit, list or remove this file.

Exit status:

0 Successful completion.

>0 An error occurred.

Caveats:

If you inadvertently enter the **crontab** utility with no argument, don't try to get out by typing Ctrl – D (end-of-file), since this would replace your crontab file with an empty file. In this case, you should exit by typing your interrupt character, which is typically Ctrl – C.

See also:

cron

ctags

Generate tags files (POSIX)

Syntax:

ctags *files*...

Options:

files The pathnames of the files that are to be scanned for tags.

Description:

The **ctags** utility generates a file called **tags** from a group of C source files.

Each C source file is scanned for **#define** statements and function definitions. The name of the macro or function becomes the name of a tag. For each tag, a line is added to the **tags** file, which contains:

- the name of the tag
- a tab character
- the name of the file containing the tag
- a tab character
- a way to find the particular line within the file

The **less**, **more**, **vi**, and **vedit** utilities can use entries in the **tags** file to locate and display a definition.

Examples:

Generate tags for all the C source and header files in the current directory:

```
ctags *.[ch]
```

© 2005, QNX Software Systems

Author:

Steve Kirkendall; **ctags** is part of the **elvis** suite.

See also:

less, **more**, **vedit**, **vi**

Syntax:

cut -c *list* **[***file***...]**

cut -f *list* **[-d** *delim***] [-s] [***file***...]**

Options:

-c *list*	Cut out the characters found in the character positions specified by *list*. For example, a *list* of **-c 1-64** would output the first 64 characters of each line.
-d *delim*	Use the delimiter specified by *delim* (default is **tab**).
-f *list*	Cut out the fields specified by *list*. For example, a *list* of **-f 2,9** would output the second and ninth fields. The fields described by *list* are assumed to be separated in the file by a delimiter character (see option **-d**). Lines without field delimiters are passed through intact, unless **-s** is specified.
-s	If **-f** is specified, suppress lines with no field delimiters.
file	The pathname of a text file, whose contents are used instead of the standard input.

Description:

For every file you name, the **cut** utility cuts out columns or fields from each line, concatenates them, and writes them to the standard output.

If the fields are of a fixed length, they can be selected by character position with option **-c**. If, however, the fields vary in length from line to line, they can be selected with option **-f**, provided they are separated by a delimiter character. By default, **cut** assumes the field delimiter character to be **tab**. You can specify another delimiter through option **-d**.

In options **-c** and **-f**, *list* is a comma-separated list of integers (in increasing order), with an optional dash (**-**) to indicate ranges.

© 2005, QNX Software Systems

You can use the **cut** utility as a filter; if no files are given, the standard input is used.

Examples:

The following are examples of the *list* argument:

***list* argument:**	**Meaning:**
1,4,7	Select the first, fourth, and seventh characters or fields.
1-3,8	Equivalent to 1, 2, 3, 8.
-5,10	Equivalent to 1, 2, 3, 4, 5, 10.
3-	Equivalent to the third through last.

Map userids to names:

```
cut -d: -f1,5 /etc/passwd
```

List filenames and their permissions:

```
ls -l | cut -c57-79,56,56,1-11
```

Files:

The input files are text files.

Exit status:

0 All input files were output successfully.

>0 An error occurred.

See also:

awk, **grep**, **join**, **paste**, **sed**

date

© 2005, QNX Software Systems

Display or set the date and time (POSIX)

Syntax:

Display date and time:

```
date [-u] [-s seconds] [+format]
date -t
```

Set date and time:

```
date [-u] date
```

Options:

-s *seconds*	Display the string equivalent of this date, supplied as seconds since the Epoch. This value is used instead of the system time value for the number of seconds.
-t	Display the current system time as the number of seconds elapsed since the Epoch (0 hours, 0 minutes, 0 seconds, January 1, 1970 Coordinated Universal Time).
-u	Perform operations using Coordinated Universal Time (UTC) instead of local time. UTC is the standard term for Greenwich Mean Time (GMT).
date	A date specification to set the date to. Only the superuser (**root**) can change the date.
+*format*	The format in which the date and time are to be displayed.

Description:

The **date** utility is used to display and set the current system date and time in software. Only the superuser (**root**) may use **date** to set the time.

Displaying the date

The **date** utility normally displays the current date and time according to QNX's internal time, maintained in software as the number of seconds since the Epoch (00:00 January 1, 1970). When the **-s** *seconds* option is used, **date** uses the value specified by the *seconds* argument instead of the current QNX time. When the **-t** option is used, **date** will print the current number of seconds since the Epoch.

You can specify the format and content of the displayed date and time with the **+***format* option. The *format* is composed of ASCII character and field descriptors, in a manner similar to a C language *printf()* format specifier (the specific characters used to specify field types are, however, completely different). In the output, each field descriptor is replaced by its corresponding value (see the following table). All other characters are copied to the output without change. To display a "**%**" character, encode it as **%%**. The **date** utility always terminates its output with a **newline** character.

Descriptor:	Will be replaced by:
%a	Locale's abbreviated weekday name.
%A	Locale's full weekday name.
%b	Locale's abbreviated month name.
%B	Locale's full month name.
%c	Locale's appropriate date and time representation.
%d	Day of the month as a decimal number (01-31).
%D	Date in the format *mm/dd/yy*.
%h	A synonym for **%b**.
%H	Hour (24-hour clock) as a decimal number (00-23).
%I	Hour (12-hour clock) as a decimal number (01-12).

continued...

© 2005, QNX Software Systems

Descriptor:	Will be replaced by:
%j	Day of the year as a decimal number (001-366).
%m	Month as a decimal number (01-12).
%M	Minute as a decimal number (00-59).
%n	Insert a **newline** character.
%p	Locale's equivalent of either am or pm.
%r	12-Hour clock time (01-12) using the AM/PM notation in the format *HH:MM:SS* **AM** or *HH:MM:SS* **PM**
%S	Seconds as a decimal number (00-61). The range of values for **%S** extends from 0 to 61 seconds to accommodate the occasional leap second or double leap second.
%t	Insert a **tab** character.
%T	24-Hour clock time (00-23) in the format *HH:MM:SS*.
%U	Week number of the year (Sunday as the first day of the week) as a decimal number (00-53).
%w	Weekday as a decimal number (0 (Sunday)-6).
%W	Week number of the year (Monday as the first day of the week) as a decimal number (00-53).
%x	Locale's appropriate date representation.
%X	Locale's appropriate time representation.
%y	Year without century as a decimal number (00-99).
%Y	Year with century as a decimal number.
%Z	Time zone name, or no characters if no time zone can be determined.

Setting the date

If you're a system administrator running as **root**, you may use **date** to set the QNX system time. To set the hardware clock to match the current system time set by **date**, you should use the **rtc** utility.

If an adjustment needs to be made, you usually first set the date to the correct setting using **date**, then run **rtc -shw** to copy the QNX system time to the battery backed-up hardware clock (on an IBM AT or PS/2).

The **date** utility recognizes three formats for setting the time:

Format 1: [[[*CC*]*YY*]*MM*]*DD*]*hhmm*[.*SS*]

Format 2: *DD* [*Month* [[*CC*]*YY* [*hh* [*mm* [*SS*]]]]] [**am|pm**]

Format 3: (valid for years 70 to 99 only)

MMDDhhmmYY

Where:

CC	the century (e.g. 19 if the year is 1997)
YY	the year modulo 100 (e.g. 97 if the year is 1997, 00 if the year is 2000)
MM	the numerical month of the year (01 for January, 02 for February, etc.)
Month	either the numerical month (1, 2,...12) or the standard English abbreviation for the month (jan, feb,...dec)
DD	the day of the month (e.g. 02 for the 2nd day)
hh	the hour of the day (e.g. 06 for 6 AM, 14 for 2 PM)
mm	the minute of the hour (e.g. 05 for five minutes past the hour)
SS	the seconds of the minute (e.g. 00 for 0 seconds)

© 2005, QNX Software Systems

am|pm literally **am** or **pm**; can be combined with hour values less than 13 if you don't want a 24-hour clock. Used only with format 2.

Format 1 is compatible with the **touch** utility. Since each field is two digits, you must specify a leading 0 for single-digit numbers. You should find this format particularly useful for adjusting the time of day, since its minimal form is just *hhmm* (hour and minute). If the optional *CC* (century) is not specified, years 00-37 will be taken to mean 2000-2038, and years 70-99 will be taken to mean 1970 to 1999. Years past 2038 cannot be expressed by the signed 32-bit number used to store date and time internally, and are not supported.

Format 2 follows the date convention used in QNX 4.00 and earlier. This format is assumed if there's more than one operand (the other two formats consist of a single string of digits), or if there's just one number that's two or less digits in length.

Format 3 follows the UNIX System V date conventions. It's similar to the Format 1, with the month and day specified, but the year is specified at the end of the specification instead of the beginning. Format 3 (with the year trailing the date/time) cannot be used to set the date if the year is 2000 or above, because of the ambiguity differentiating between format 3 and format 1 if the year is 00 through 12.

If only one operand is supplied and it is greater than two characters in length, **date** will interpret the date as being format 1 or format 3. If multiple operands are supplied or a single operand of 2 characters or less is given, then the date will be interpreted as being in format 2. Only in the case of a 10-character date with the last two digits being greater than 59 (an invalid minute, if format 1 is used) will the date be interpreted as format 3. In all other cases format 1 (POSIX **touch**-style) will be used.

Examples:

The following are input/output examples of **date** used at arbitrary times in the POSIX locale:

Input: `date "+DATE: %m/%d/%Y%nTIME: %H:%M:%S"`

Output:
```
DATE: 11/21/1997
TIME: 13:36:16
```

Input: `date "+TIME: %r"`

Output:
```
TIME: 01:36:32 PM
```

The following are examples of **date** used to set the date and time:

Set the system date/time to 9:00 AM local time, March 10 2002:

```
date 200203100900.00
```

or, you can specify just the last two digits of the year:

```
date 0203100900.00
```

Set the system date to March 10, 2002, without changing the time:

```
date 10 3 2002
```

Adjust the system time to 4:34 pm local time without changing the date:

```
date 1634
```

The following command, which illustrates the use of **date -s**, lets you see a text representation of the date of the last entry in the **/usr/adm/syslog** file (in this file, the first column of each record is the (long) time in seconds since the Epoch):

Input:
```
date -s `tail -n1 /usr/adm/syslog | \
cut -f1 -d ' '`
```

Output:
```
Wed Apr 15 14:25:49 EDT 1997
```

For more information, see the **cut**, **logger**, and **tail** utilities.

© 2005, QNX Software Systems

Environment variables:

TZ Specifies the local time zone. The value of **TZ** affects the conversion between the system clock (UTC) and the local time.

Exit status:

0 The date was displayed or set successfully.

>0 An error occurred.

Caveats:

Some field descriptors are of unspecified format when not in the POSIX locale. As a result, parsing the output of **date** may be difficult in other locales.

The system date cannot be set beyond January of 2038 which is when the signed 32-bit value used to store the seconds since the Epoch will reach its maximum value. If you need to embed QNX in a product which must operate past 2038 without maintenance or modification, contact QNX Software Systems for advice.

Different versions of the **date** utility shipped with various releases of QNX have differing behavior with two-digit years representing years past 2000. To be sure of correct operation with all versions, avoid format 3 (Sys V format, with the year trailing the date/time), and use four digits when specifying years.

A product bulletin on QNX and year 2000 issues is maintained on the QNX Software Systems homepage. (Search for **year 2000**.)

See also:

clock, **rtc**

Year 2000 product bulletin

dcheck

Check a disk for bad blocks (QNX)

Syntax:

dcheck [*options*] *drive*

Options:

-b *blk_cnt*	The maximum number of blocks to check.
-B *max_blks*	The maximum number of blocks to read at a time; can be up to 32 (the default).
-e	If possible, disable drive error correction during check for bad blocks.
-f *first_blk*	The first block to check.
-l	("el") Loop until input (switching serial/random).
-L *loops*	Loop as in the **-l** option, but with a specified number of loops.
-m	In the bitmap, mark bad blocks as unavailable.
-p	Prompt before starting.
-q	Be quiet; don't display progress information.
-r	Use a random head movement algorithm.
-v	Be verbose; display every bad block on the disk.
-V	Verify write after read. (Filesystem caching effects can render this option to be of little value.)
-w	Write after read (non-destructive) check.
drive	The name of the disk (e.g. **/dev/fd0**, **/dev/hd0t77**) or the root of the filesystem.

Description:

The **dcheck** utility verifies that a disk has been correctly formatted by attempting to read every block on the drive. The block numbers of any blocks that cannot be read are displayed (in hex) to standard output. A summary of the total number of bad blocks is also displayed. You can use **dcheck** to check any formatted disk, including disks that contain files. The files won't be damaged.

If the number of blocks to verify is not specified, **dcheck** will obtain this information from the filesystem and will check *all* the blocks on the specified drive.

If a disk has been initialized for QNX, you should use the **-m** option to remove any bad blocks from the disk allocation bitmap (**/.bitmap**). This is especially true for hard disks. When you specify the **-m** option, **dcheck** attempts to read the file **/.bad_blks** from the disk. This file contains a list of all known bad blocks, in sorted order. If **/.bad_blks** is found, **dcheck** will read it, and when it is finished checking the disk, **dcheck** will update the bitmap and recreate the **/.bad_blks** file. Note that the **dcheck** utility only adds to, but never removes, bad block information in this file.

Some blocks may be marginal, so if you run **dcheck** multiple times (see the **-l** and **-L** options), you can increase the chance of these blocks being recognized and added to the **/.bad_blks** file.

The **/.bad_blks** file is also recognized by the **chkfsys** utility.

To help you find any marginal blocks, **dcheck** has a number of options to provide additional checking of a disk. For example, the **-r** option checks the blocks in a random order-each check consists of a random number of blocks between 1 and 32 (or less, depending on the value specified in the **-B** option). The **dcheck** utility keeps track of the checked blocks and checks each one only once. This option allows you to find blocks that are bad due to a slight lag time in head movement.

The **-l** option will continuously check the disk until you stop it. For this option, **-r** is implicitly used and is toggled for each invocation. That is, for the first loop, random checking is set on; for the second loop, it is off, etc. At the end of each complete check, you are prompted to stop the loop. If you don't stop it within 15 seconds, **dcheck** will be started again, etc. The **-L** option is identical to **-l** with an upper limit to the number of loops.

The **-w** option will rewrite each block on the device after reading it. This is a nondestructive check that tests the write portion of the hardware. Note that, although this is a more thorough test, you must pay for it with additional time that is dependent on the hardware.

The **-V** option is similar to the **-w** option, in that **dcheck** rewrites each block after reading it but in this case **dcheck** also rereads each block after the rewrite check and compares this second read with the first. Like the **-w** option, this test is also nondestructive. Note, however, that this is a more thorough test that takes longer.

If you wish to check a disk on another node, you must indicate the disk explicitly via a **//node** prefix.

The **-e** option requests that error correction (where supported) not be used. This lets **dcheck** detect marginal blocks as if they were bad blocks. Hence, specifying the **-e** option in combination with the **-m** option — which marks bad blocks as unavailable-removes marginal blocks from the disk allocation bitmap. Note that error correction is re-enabled after **dcheck** terminates.

Examples:

Check all blocks on the hard disk and mark bad blocks in the bitmap:

```
dcheck -m /
```

Check the first 640 blocks on the floppy disk:

```
dcheck -b 640 /dev/fd0
```

Check all blocks on the hard disk of node 3:

```
dcheck //3/dev/hd0t77
```

Files:

The **dcheck** utility will utilize the standard input for reading user responses to prompts. The user will be prompted only if the **-p** (pause before starting) option or **-l** (loop until input) options are used.

Messages will be written to standard error when internal errors occur which might prevent a complete check, or when **dcheck** discovers bad blocks on the disk. Information not related to errors is written to the standard output.

All *drive* parameters specified on the command line must specify block special files.

If the **-m** (mark bad blocks) option is specified, the block special file being checked must be a currently mounted QNX partition. The **.badblocks** and **.bitmap** files on that filesystem will be updated if **dcheck** discovers any bad blocks.

Exit status:

0 No bad blocks were found.

>0 An error occurred or bad blocks were found.

Caveats:

The **dcheck** utility normally opens the disk in read-only mode. However, if you specify the **-m**, **-w**, or **-V** options, the disk is opened in read/write mode. For read/write access, there must be no open files on the disk, or **dcheck** will fail with a “Device or resource busy” message. While **dcheck** is working in read/write mode, no other utilities or programs will be allowed to access the disk.

When using the **-m** option, if **dcheck** is terminated by a SIGBREAK or other signal, any pending bad blocks may not be recorded. In any event, the results are nondestructive.

Filesystem caching may render some options, such as **-V** to be of little value. In the case of **-V**, the read after write may simply read the data back out of the cache instead of from the disk media. Another cache effect to watch out for is that bad blocks are 'sticky' in the cache. Even though the failure might be intermittent, it will not be possible for force **Fsys** to re-read the block as long as it remains in cache. It will always return the block as bad.

See also:

chkfys, **dinit**, **fdformat**, **Fsys**

dd

© 2005, QNX Software Systems

Convert a file while copying it (UNIX)

Syntax:

dd [if=*input_file***] [of=***output_file***] [***options***]**

Options:

All numbers can be followed by a multiplier:

b	blocks (×512)
k	kbytes (×1024)
w	words (×2)
x*m*	multiply by *m*

if=*input_file*	Read from *input_file* instead of the standard input.
of=*output_file*	Write to *output_file* instead of the standard output. Unless **conv=notrunc** is given, truncate the file to the size specified by **seek=** (0 bytes if **seek=** is not given).
ibs=*bytes*	Read *bytes* bytes at a time.
obs=*bytes*	Write *bytes* bytes at a time.
bs=*bytes*	Read and write *bytes* bytes at a time. Override **ibs** and **obs**.
cbs=*bytes*	Convert *bytes* bytes at a time.
skip=*blocks*	Skip *blocks* **ibs**-sized blocks at start of input.
seek=*blocks*	Skip *blocks* **obs**-sized blocks at start of output.
count=*blocks*	Copy only *blocks* **ibs**-sized input blocks.

conv=*conversion*[**,***conversion*...]

Convert the file as specified by the *conversion* arguments. Conversions are:

ascii	Convert EBCDIC to ASCII.
ebcdic	Convert ASCII to EBCDIC.
ibm	Convert ASCII to alternate EBCDIC.
block	Pad newline-terminated records to size of **cbs**, replacing newline with trailing spaces.
unblock	Replace trailing spaces in **cbs**-sized block with newline.
lcase	Change uppercase characters to lowercase.
ucase	Change lowercase characters to uppercase.
swab	Swap every pair of input bytes. Unlike the Unix **dd**, this works when an odd number of bytes are read. If the input file contains an odd number of bytes, the last byte is simply copied (since there is nothing to swap it with).
noerror	Continue after read errors.
notrunc	Do not truncate the output file.
sync	Pad every input block to size of **ibs** with trailing NULs.

Description:

This manual page documents the GNU version of **dd**. The **dd** utility copies a file (from the standard input to the standard output, by default) with a user-selectable blocksize, while optionally performing conversions on it. It is meant for writing raw data directly to devices such as tape and disk or writing over the network, with control over blocking factors and character set translations.

© 2005, QNX Software Systems

This command may be used for copying partial files. The block size, skip count, and number of blocks to copy can be specified. Where sizes are expected, they are in bytes. However, the letters **w**, **b**, or **k** may be appended to the number to indicate words (2 bytes), blocks (512 bytes), or K (1024 bytes), respectively. When **dd** is finished, it reports the number of full and partial blocks read and written.

Examples:

Copy file **file1** to **file2**, converting all text to lowercase letters:

```
dd if=file1 of=file2 conv=lcase
```

Exit status:

>0 An error occurred.

0 The copy/translate operation was successful.

Author:

GNU

See also:

cat, **cp**, **pax**, **tr**, **head**

Debugger32

Low-level 32-bit system debugger (QNX)

Syntax:

Debugger32 [-b *baud***] [-d** *text***] [-s** *port***]**

Options:

-b *baud* If **-s** is specified, set the baud rate to *baud* (default is 9600).

-d *text* Dial a modem by outputting *text*. An OR bar (**|**) in *text* is mapped to a carriage return. The debugger will wait for an **Esc** to continue.

-s *port* Send output to serial port. (Default: 0x3f8)

-t Set serial port to 0x3f8 at 9600 baud.

-z*config_option...*

Set the configuration option specified. Valid configuration options are:

u|m	Unix (**u** or WASM (**m** assembly format
b|h|l	Dump in bytes (**b**), halfwords (**h**) or longs (**l**).
q|v	Quiet (**q**) or verbose (**v**). In verbose mode, **Debugger32** will show registers on entry.
s|k	Use serial port (**s**) or system console/keyboard (**k**).
e|Q	Use Unix/EMACS line editing (**e**) or use QNX line editing (**Q**).

These options may be set from within the debugger through the colon (**:**) command.

Description:

The **Debugger32** utility is a low-level debugger that lets you set breakpoints and single-step through programs, display and edit

memory, disassemble code, and examine I/O ports. You can use this debugger for debugging interrupt handlers.

This debugger will function only with versions of **Proc** released with QNX 4.23 or later.

The debugger operates *below* the kernel, thus allows debugging the kernel, interrupt and fault routines.

This debugger must be built into the operating system image; it cannot be started from the command line. You always place **Debugger32** in the last line of the build file. For example:

```
sys/Debugger32
$ Debugger32
```

For more information, see the **buildqnx** utility.

Unless you specify the **-D** option to **Proc**, the debugger will go through two checkpoints when the operating system boots. You'll be prompted:

1. as soon as **Proc** can register the debugger
2. at the point when all built-in processes are scheduled and ready to run-this is just before they start running. Here you may set breakpoints on processes. (You may also wish to use the debugger's **l** *pid* command to set the LDT to a specific process.)

You can invoke this debugger with the following key chord: Ctrl – Alt – Esc.

WARNING:

Don't use this debugger in a multiuser environment, because it disables interrupts and freezes the entire system.

The debugger operates in a simple line-edited mode (most familiar editing characters are available) accepting commands of the form

cmd opt-arguments

Most commands are single letter, some with modifiers. For example **d** displays memory, **dw** display memory in word-length quantities. Most arguments are expressions, which have a common syntax. The expression evaluator is crude, with cryptic error messages, and expressions may not contain space characters.

The command set supported by the debugger in QNX 4.23 and later is different from the one supplied with QNX versions 4.00 through 4.22.

Command mode

Any the debugger command prompt (*****), the following are valid:

Command	Arguments	Description
d*size*	[*expr0* [*expr1*]]	display memory at *expr0* for *expr1* bytes.
D	*expr*	disassemble instructions at the address *expr*, enter disasm-mode.
i*size*	*expr*	read from I/O location *expr*.
o*size*	*expr0 expr1*	write *expr1* to I/O location *expr0*.
e*size*	*expr*	edit memory at location *expr*.
r*register*	[*expr*]	display or set register to *expr*.
a	*expr*	show page table entry for address *expr*.
.	*expr*	print segment table entry for selector *expr*.
/p	*expr*	display memory as a process table entry *expr*.

continued...

Command	Arguments	Description
/P	*expr*	display proc table entry for pid *expr*.
/r	*expr*	display memory at *expr* as a saved register set.
/R	*expr*	display saved register set for pid *expr*.
/m	*expr0* [*expr1*]	display mxfer_entry at *expr0* (for *expr1*)
?		show registers and fault.
"		comment till end of line.
!	*expr*	print expression *expr*.
b	*expr*	set breakpoint at address *expr*.
b	**?**	show all breakpoints.
g		continue execution
s		enter single-step mode.
v		continue from fault via “stored” fault vector.
V	[*expr*]	trap fault *expr* (default all)
u	[*expr*]	clear breakpoint at address *expr*, or clear all breakpoints.
U	[*expr*]	restore fault *expr* (default all)
=		continuing, keeping breakpoint

Single step mode

In single step mode the prompt is the disassembled opcode to be executed. If the user presses Enter it will not be executed. Space, i or I will execute one opcode and re-enter single-step mode. If the opcode is a “call”, the O key will step over the function.

Disassemble mode

Disassemble mode (**D**) shows each opcode, Space shows the next one. Any other key moves back to command mode.

Debugger expressions

The debugger parses expressions that conform to the following grammar. The precedence of operations are as in C.

```
expr
  : primary
  | '(' expr ')'
  | expr '*,/,%' expr
  | expr '+,-' expr
  | expr '<<,>>' expr
  | expr '==,!=' expr
  | expr '&' expr
  | expr '^' expr
  | expr '|' expr
  | expr '&&' expr
  | expr '||' expr
  ;
primary
  : value
  | primary ':' primary
  | register
  | '!' primary
  | '~' primary
  | '-' primary
  | '*' primary
  ;
value
  : (0x)?'0-9a-f' + ;
register
  : eax|ebx|ecx|edx|esi|edi|ebp|esp|eip|flags
  |  ax| bx| cx| dx| si| di| bp| sp| ip
  |  cs| ds| es| ss| fs| gs
  | cr[0-3]
  | dr[0-7]
  ;
```

© 2005, QNX Software Systems

Line editing

The debugger supports limited line editing. The default behavior is EMACS-mode, but can be switched to QNX-mode by the *colon* commands. This table summarizes the editing characters.

Edit Cmd	EMACS-mode	QNX-mode	Description
home	Ctrl – A	Home	Move cursor to beginning of line.
end	Ctrl – E	End	Move cursor to end of line.
right	Ctrl – F	→	Move cursor right one character.
word right	ESC f	*none*	Move cursor right one word.
left	Ctrl – B	←	Move cursor left one character.
word left	ESC b	*none*	Move cursor left one word.
line kill	Ctrl – U	*none*	Erase entire line.
eol kill	ESC w	*none*	Erase from cursor to end of line.
char delete	Ctrl – H	backspace	Delete current char.
word delete	Ctrl – H	*none*	Delete current word.
previous	Ctrl – P	↑	Recall previous line.
next	Ctrl – N	↓	Recall next line.

Examples:

Dial the specified phone number before starting the debug session:

```
Debugger32 -s 3f8 -d ATDT5910941
```

See also:

buildqnx, **Proc**

WATCOM Debugger User's Guide

Dev

© 2005, QNX Software Systems

Device manager (QNX)

Syntax:

Dev [-n *max_devices***] [-N** *prefix***] &**

Options:

-n *max_devices*

The maximum number of terminal devices to support (default is 64).

-N *prefix* The directory name to register in the pathname space. (default is **/dev**)

Description:

The **Dev** manager is responsible for managing all *terminal devices* in a QNX system.

Once **Dev** is started, the driver processes may be started. The drivers register themselves with **Dev**, making the hardware they control available to the system as additional devices. Some of the common drivers shipped with QNX are:

Driver	Description
Dev.ansi	console I/O (ANSI terminal emulation is default; also supports QNX terminal emulation)
Dev.con	console I/O (QNX terminal emulation is default; also supports ANSI terminal emulation)
Dev.par	parallel I/O
Dev.pty	pseudo tty devices
Dev.ser	serial I/O

Dev always supports the NULL device (called **/dev/null**); this device always takes up one of the device entries (**-n** option).

The **Dev** manager must be run as **root**.

Examples:

Start **Dev**; use defaults:

```
Dev &
```

Allow 100 devices:

```
Dev -n 100 &
```

Files:

Dev closes its standard input, standard output, and standard error immediately upon startup.

Dev adopts the directory **/dev** in the pathname prefix tree of the node it is run on. The special NULL device, **/dev/null** will also appear.

Exit status:

Dev does not normally terminate. Because of its importance in the system (especially a standalone one), **Dev** will choose to reduce its memory requirements by reducing the number of devices it will support instead of terminating with an error. However, there are still cases where **Dev** will exit, such as in the case of an invalid command-line parameter, failure to allocate internal structures sufficient for the support of at least a single device, and failure to attach the **/dev** prefix with **Proc**.

>0 An error occurred in startup.

© 2005, QNX Software Systems

See also:

Dev.*, **modem**, **Mouse**, **stty**, **tinit**

Installation & Configuration

Dev.ansi

ANSI console driver (QNX)

Syntax:

Dev.ansi [-C *number***] [-H] [-I** *number***] [-k** *rate***[,**_delay_**]]**
[-L -N *name***] [-n** *number***]**
[-O *number***] [-Q] &**

Options:

-A Start in ANSI terminal emulation mode (default).

-C *number* Set the size of the canonical input buffer (default is 256 bytes).

-h Enable hardware scrolling.

-H Disable hardware scrolling. (Default)

-I *number* Set the size of the raw input buffer (default is 256 bytes).

-J Disable Japanese keyboard mapping.

-k *rate*[**,**_delay_]

Set the keyboard rate in Hz (default is 30) and the delay in milliseconds (default is 500).

To prevent **Dev.ansi** from setting the keyboard rate to the default value of 30 Hz, specify **-k 0**. This may be necessary with some keyboards.

-K No keyboard. Typically used (sometimes in conjunction with **-S**) to manage virtual consoles which may be manipulated with **ditto**, even though no physical keyboard is available.

-L [**P**][**N**][**C**][**S**]

Set the initial state of the keyboard and its LEDs:

- **P** — preserve the keyboard state. This overrides all the other **-L** options.

- **N** — turn NumLock on.
- **C** — turn CapsLock on.
- **S** — turn ScrollLock on.

-m *hex_number*
Value used to set VGA misc register on console switch.

-n *number* Allow no more than this number of virtual consoles (default is 1, maximum 10).

-N *name* Use this prefix for the device name. If a full path (i.e. one which includes the directory under which the device should be registered) is not specified, the name will implicitly be registered under **/dev**. (default is **/dev/ansi**)

-O *number* Set the size of the output buffer (default is 2048 bytes).

-Q Start in QNX terminal-emulation mode (default is ANSI).

-S No screen. Typically used in conjunction with **-K** to manage virtual consoles which may be manipulated with **ditto**, even though no video adapter is installed in the machine.

Description:

The **Dev.ansi** driver manages console devices (i.e. text mode video and the system keyboard) in a QNX system. **Dev.ansi** may not be started until the Device Manager (**Dev**) has been started.

Use of hardware scrolling can greatly improve the speed at which text mode consoles can scroll. To enable hardware scrolling, specify the **-h** option. When **-h** is specified, if **Dev.ansi** detects that the video hardware can support hardware scrolling, it programs the hardware to do so. Using hardware scrolling and running programs that are

sensitive to the state of the video hardware can result in interference with the program's video output. In such cases, disable hardware scrolling.

Keyboard control

Some functions of **Dev.ansi** are (or can be) controlled by keyboard keychords. These are:

- Switching the visible console from one virtual console to another
- Change fonts on the currently visible virtual console
- Invoke the low-level debugger (if the debugger is installed)
- Rebooting the computer

Switching between virtual consoles

Each virtual console can be running different applications that use the entire screen. The keyboard is attach to the virtual console that's currently visible. You can switch from one virtual console to another — and thus from one application to another — by entering the following keychords:

If you want to see:	Press:
the next active console	Ctrl – Alt – Enter *or* Ctrl – Alt – + (plus)
the previous active console	Ctrl – Alt – - (minus)

The + (plus) and - (minus) keys used in the console-switching keychords are those found on the numeric keypad.

You can also jump to a specific console by using the Ctrl – Alt – *n*, where *n* is a numeric digit that represents the console number of a virtual console. For instance:

If you want to see:	Press:
/dev/con1	Ctrl – Alt – 1
/dev/con2 (if available)	Ctrl – Alt – 2
⋮	⋮
/dev/con10 (if available)	Ctrl – Alt – 0

You can disable keyboard console switching with the **stty +noswitch** command.

Changing the console fonts

If you are using standard VGA or EGA compatible video hardware, the console driver supports a variety of screen fonts. On all video hardware, the driver defines font 0 as a 80×25 text font. With a VGA-type adapter, an additional 80×50 font is defined as font 1. On an EGA adapter font 1 is defined as a 80×43 text font.

To load new fonts into **Dev.ansi**, use the **cfont** utility. You can use this utility to provide fonts of different sizes or fonts that contain alternate character sets. Up to four fonts may be loaded, which includes the two fonts which are defined by default. The **cfont** utility may be used to supplant the default font 0 and font 1 with alternate fonts, in addition to loading fonts 2 and 3.

To change the font associated with the current console (which may also change the console *columns*×*rows*), use the following keychords:

To select:	Press:
the next font (up to 3)	Ctrl – Alt – **>**
the previous font (down to 0)	Ctrl – Alt – **<**

The driver keeps track of the font being used by each virtual console. All consoles initially display font 0. You can disable keyboard font changing with the **stty +noresize** command.

Rebooting the computer

To reboot the computer, you can use this keychord:

Ctrl – Alt – Shift – Del

You can disable the ability to reboot the computer via the keychord by using the **`stty +noboot`** command.

CAUTION: Before entering the keychord to reboot the computer, make sure that no applications or utilities are running on your computer. If applications are running, they will be terminated in mid-process and may not leave their files in a state from which they can be recovered when the system comes up again. When the keychord is used to reboot the computer, the filesystem and other key resources will continue to run for a short time (typically 10 seconds or so) during which they will attempt to clean up open files and flush pending data to disk. It is possible that this period of time will be insufficient, in which case data may be lost and the filesystem will require maintenance using **`chkfsys`**. It is safest to ensure that applications have been shut down cleanly and the system is idle before rebooting. (See the **`shutdown`** utility.)

Invoking the low-level debugger

If the low-level debugger (**`Debugger32`**) is running, you can invoke it with this keychord:

Ctrl – Alt – Esc

© 2005, QNX Software Systems

WARNING: When the debugger is invoked, interrupts are disabled and the entire system will be frozen. Use this keychord with caution. Read the documentation for `Debugger32` before using.

You can disable the ability to invoke the low-level debugger with the keyboard keychord by using the **`stty +nodebug`** command.

Terminal-emulation modes

When **`Dev.ansi`** is started, it makes the terminal type the same for all its consoles. By default, the consoles will all emulate the ANSI terminal. The **`Dev.ansi`** driver currently has the ability to emulate the historic **`QNX`** terminal type. If **`-Q`** is specified, they will all emulate the QNX terminal. If **`Dev.ansi`** is started as **`Dev.con`**, the default becomes QNX terminal emulation and **`-A`** should be used to override the default and use ANSI emulation. (See **`Dev.con`**.)

When users of the system are predominantly console users, it may prove convenient to modify the **`/etc/default/login`** file to set the **TERM** environment variable to the terminal type the driver is emulating. Some OS images (made by **`buildqnx`**) have set the **TERM** environment variable on the **`sinit`** command line.

Each console keeps track of its terminal emulation mode. To change the emulation mode of a particular console, run the following commands on that console:

To change to:	Run these commands:
QNX emulation	`export TERM=qnx` `stty protocol=0 term=$TERM`
ANSI emulation	`export TERM=ansi` `stty protocol=1 term=$TERM`

The ANSI control codes and keyboard sequences are defined in this manpage. The control codes and keyboard codes for the QNX mode of this driver are listed in the man page for **Dev.con**.

CAUTION:

Unlike the **TERM** environment variable, **stty term=** settings aren't restored when a shell script terminates. Therefore, if the above pairs of commands are run in a shell script, the **stty** settings must either be reloaded before exiting from the shell script or reloaded from the command line after the script terminates.

Character sets

The **Dev.ansi** driver, in ANSI mode, allows the character sets in use to be chosen from a "palette" of character sets, each of which is independently programmable to contain one of several builtin character sets.

The in-use range of characters is divided into four regions which span character numbers (in hexadecimal) 0x00 through 0xff. Two of these regions are fixed sets of control characters, while the other two are configurable to contain a choice of character sets:

Hex:	Name:	May be set to one of:
0x00-0x1f	C0 (Control Zero)	Not configurable
0x20-0x7f	GL (Graphics Left)	G0, G1, G2, G3
0x80-0x9f	C1 (Control One)	Not configurable
0xa0-0xff	GR (Graphics Right)	G1, G2, G3

© 2005, QNX Software Systems

Mapping the GL and GR character sets

Each of the GL and GR in-use character sets can be set to a choice of several character sets from the G0, G1, G2 and G3 character sets. (For configuring these, see the section entitled "Setting the G0, G1, G2 and G3 Character Sets".)

The screen control codes to set GL and GR are as follows:

To set:	To:	Use:
GL	G0	**{**LS0**}** = **{**SI**}** (0f)
GL	G1	**{**LS1**}** = **{**SO**}** (0e)
GL	G2	**{**LS2**}** = **{**ESC n**}** (1b 6e) or **{**SS2**}** (8e)
GL	G3	**{**LS3**}** = **{**ESC o**}** (1b 6f) or **{**SS3**}** (8f)
GR	G1	**{**LS1R**}**= **{**ESC ~**}** (1b 7e)
GR	G2	**{**LS2R**}**= **{**ESC **}}** (1b 7d)
GR	G3	**{**LS3R**}**= **{**ESC **|}** (1b 7c)

The **{**LS***}** codes stand for "Locking Shift". When character sets are selected by these means, they remain in effect until another **{**LS***}** code is sent.

The **{**SS***}** codes stand for "Single Shift" and will affect the next character only. After that character, the character set in effect reverts to its previous setting. There are only two **{**SS***}** codes, **{**SS2**}** and **{**SS3**}** which maps G2 into GL and G3 into GL, respectively.

Setting the G0, G1, G2 and G3 character sets

The G0 through G3 characters sets may each be set to any of the available builtin fonts. The control code to do this is:

ESC *g s*

Where:

g:	Sets:
(	G0
)	G1
*****	G2
+	G3

And:

s:	Specifies:
B	ASCII
0	Special (DEC Graphic)
<	ISO-Latin1 Supplemental
U	PC Character Set

Character set defaults

In-use char set:	Defaults to:
GR	G2
GL	G0

Char set:	Defaults to:
G0	ASCII charset
G1	Special charset (DEC Graphic)
G2	Supplemental charset (ISO-Latin 1)

continued...

Char set:	Defaults to:
G3	Special charset (DEC Graphic)

Character set example

Set the GL in-service character set (0x20-0x7f) to the PC character set through G1, write some characters, then switch GL back to G0:

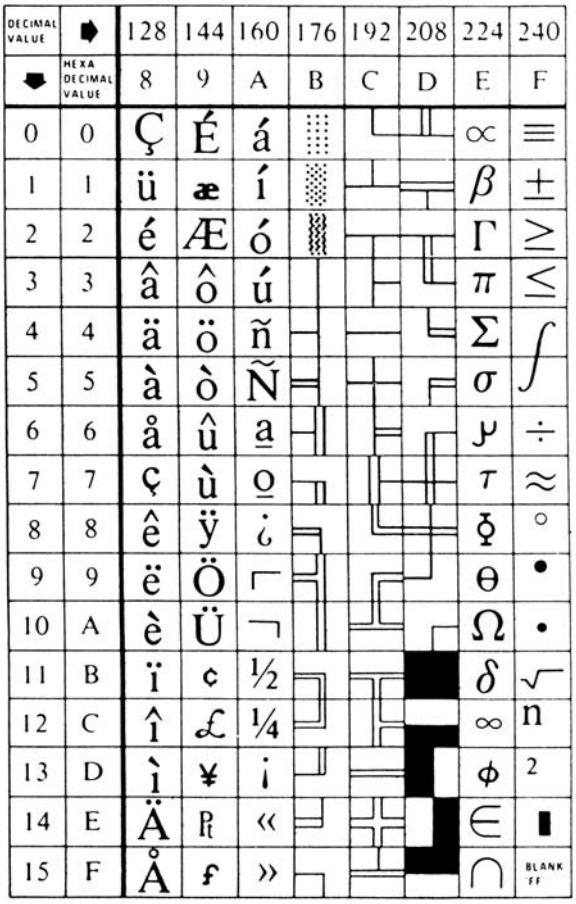
```
{ESC )U}  1e 29 55    (Set G1 to be the PC character set)
{SO}      0e          (Set GL to G1)
.
.                     (Write chars in PC graphics char set)
.
{SI}      0f          (Set GL to G0)
```

The PC character set 0x00-0x7f

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112
⬇	HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	–	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

© 2005, QNX Software Systems

The PC character set 0x80-0xff



ANSI screen control codes

Note the following abbreviations used in the tables:

(220+)	A VT220 Level 2 function
(NA)	Not ANSI standard
(NI)	Not implemented
(NFI)	Not fully implemented

C0 control codes

ASCII	ANSI Mnemonic	Hex	Action
{NUL}		00	Null
{BEL}		07	Bell
{BS}		08	Back Space (VT100 defaults to no wrap from left margin)
{HT}		09	Horizontal Tab (VT100 defaults to no autowrap)
{LF}		0A	Linfeed or Newline
{VT}		0B	Same as LF
{FF}		0C	Clears Screen (QNX Extension)
{CR}		0D	Move cursor to left margin
{SO}	{LS1}	0E	GL is set to G1
{SI}	{LS0}	0F	GL is set to G0 (default)
{XON}	{DC1}	11	XON

continued...

© 2005, QNX Software Systems

ASCII	ANSI Mnemonic	Hex	Action
{XOFF}	{DC0}	13	XOFF
{CAN}		18	Cancels ESC sequence
{SUB}		1A	Cancels ESC sequence and prints ?
{ESC}		1B	Start of ESC sequence
{DEL}		7F	Ignored on output

ESC control sequences

String	Hex	Action
{ESC 7}	1B 37	Save cursor
{ESC 8}	1B 38	Restore cursor
{ESC =}	1B 3D	Set application keypad mode
{ESC >}	1B 3E	Set numeric keypad mode (default)
{ESC D}	1B 44	7-bit codes for {IND} (84)
{ESC E}	1B 45	7-bit codes for {NEL} (85)
{ESC H}	1B 48	7-bit codes for {HTS} (88)
{ESC M}	1B 4D	7-bit codes for {RI} (8D)
{ESC N}	1B 4E	7-bit codes for {SS2} (8E)
{ESC O}	1B 4F	7-bit codes for {SS3} (8F)
{ESC P}	1B 50	7-bit codes for {DCS} (90)
{ESC [}	1B 5B	7-bit codes for {CSI} (9B)
{ESC \}	1B 5C	7-bit codes for {ST} (9C)
{ESC]}	1B 5D	7-bit codes for {OSC} (9D)

continued...

String	Hex	Action
{ESC ^}	1B 5E	7-bit codes for {PM} (9E)
{ESC _}	1B 5F	7-bit codes for {APC} (9F)
{ESC Z}	1B 5A	Identify terminal
{ESC c}	1B 63	Hard Reset (clears screen) (use {CSI ! P} for soft reset)
{ESC n}	1B 6E	(LS2) GL is set to G2 (220+)
{ESC o}	1B 6F	(LS3) GL is set to G3 (220+)
{ESC |}	1B 7C	(LS3R) GR is set to G3 (220+)
{ESC }}	1B 7D	(LS2R) GR is set to G2 (220+) (default)
{ESC ~}	1B 7E	(LS1R) GR is set to G1
{ESC sp F}	1B 20 46	Keyboard generates 7-bit C1 codes (incl. CSI) (default)
{ESC sp G}	1B 20 47	keyboard generates 8-bit C1 codes (incl. CSI) (220+)
{ESC (0}	1B 28 30	Set G0 to special charset
{ESC (<}	1B 28 3C	Set G0 to supplemental charset
{ESC (A}	1B 28 41	Set G0 to U.K. charset (Not implemented; same as ASCII)
{ESC (B}	1B 28 42	Set G0 to ASCII charset (default)
{ESC (U}	1B 28 55	Set G0 to PCterm graphics
{ESC) 0}	1B 29 30	Set G1 to special charset (default)
{ESC) <}	1B 29 3C	Set G1 to supplemental charset
{ESC) A}	1B 29 41	Set G1 to U.K. charset (NI; same as ASCII)

continued...

String	Hex	Action
{ESC) B}	1B 29 42	Set G1 to ASCII charset
{ESC) U}	1B 29 55	Set G1 to PCterm graphics
{ESC * 0}	1B 2A 30	Set G2 to special charset (220+)
{ESC * <}	1B 2A 3C	Set G2 to supplemental charset (220+) (default)
{ESC * B}	1B 2A 42	Set G2 to ASCII charset (220+)
{ESC * U}	1B 2A 55	Set G2 to PCterm graphics
{ESC + 0}	1B 2B 30	Set G3 to special charset (220+) (default)
{ESC + <}	1B 2B 3C	Set G3 to supplemental charset (220+)
{ESC + B}	1B 2B 42	Set G3 to ASCII charset (220+)
{ESC + U}	1B 2B 55	Set G3 to PCterm graphics

C1 control characters (220+)

Any 8-bit C1 code can be done with 7-bit **ESC** followed by the 8-bit code minus 0x40 hex. For instance, the CSI (control sequence introducer) can be represented in 8-bit mode as 0x9b, while in 7-bit mode it must be expressed as **ESC [** (0x1b 0x5b).

ASCII	Hex	Action
{IND}	84	move cursor down with scroll
{NEL}	85	move to left margin on next line with scroll
{HTS}	88	Set horizontal tab

continued...

ASCII	Hex	Action
{RI}	8D	move cursor up with scroll
{SS2}	8E	GL is set to G2 for 1 character
{SS3}	8F	GL is set to G3 for 1 character
{DCS}	90	Start of Device control string
{CSI}	9B	Control sequence introducer
{ST}	9C	End of Device control string
{OSC}	9D	Operating System Command
{PM}	9E	Privacy Message
{APC}	9F	Application Program Command

CSI control sequences

In 7-bit mode, CSI is **ESC [**. In 8-bit mode, CSI is (hex)0x9B.

ASCII	Hex	Action
{CSI [*n*] @}	9B [*n*] 40	Insert *n* spaces at cursor (default = 1 space)
{CSI [*n*] A}	9B [*n*] 41	Cursor up *n* rows, no wrap (default = 1 row)
{CSI [*n*] B}	9B [*n*] 42	Cursor down *n* rows, no wrap (default = 1 row)
{CSI [*n*] C}	9B [*n*] 43	Cursor right *n* columns, no wrap (default = 1 column)

continued...

© 2005, QNX Software Systems

ASCII	Hex	Action
{CSI [*n*] D**}**	9B [*n*] 44	Cursor left *n* columns, no wrap (default = 1 column)
{CSI [*n*] F**}**	9B [*n*] 46	Cursor up *n* rows, positioned in first column (default = 1 row)
{CSI [*n*] G**}**	9B [*n*] 47	Move cursor to column *n* (default = column 1)
{CSI [*r*[;*c*]] H**}**	9B [*r* [3B *c*]] 48	Cursor position (default = row 1; column 1)
{CSI [*n*] J**}**	9B [*n*] 4A	Erase 0=cur-EOS 1=HOME-cur 2=screen (default = 0 (to end of screen))
{CSI [*n*] K**}**	9B [*n*] 4B	Erase 0=cur-EOL 1=BOL-cur 2=line (default = 0 (to end of line))
{CSI [*n*] L**}**	9B [*n*] 4C	Insert *n* lines (default = 1 line)
{CSI [*n*] M**}**	9B [*n*] 4D	Delete *n* lines (default = 1 line)
{CSI [*n*] P**}**	9B [*n*] 50	Delete *n* chars (default = 1 char)
{CSI [*n*] S**}**	9B [*n*] 53	Scroll forward *n* lines (default = 1 line)

continued...

ASCII	Hex	Action
{CSI [*n*] T**}**	9B [*n*] 54	Scroll backward *n* lines (default = 1 line)
{CSI [*n*] X**}**	9B [*n*] 58	Erase cur for *n*-1 chars (default = 1 (0 chars))
{CSI Z**}**	(9B 5A)	Back tab
{c CSI [*n*] b**}**	c 9B [*n*] 62	Repeat GR or GL character *c*, *n* times. *c* is the last displayable character; *n* defaults to 1 time.
{CSI 0 c**}**	(9B 30 63)	Primary device attrib request
{CSI [*n*] d**}**	9B [*n*] 64	Move cursor to line n (default = line 1)
{CSI [*n*] g**}**	9B [*n*] 67	Tab clear 0=cursor 2=all (default = 0)
{CSI [*n*[;*n*]...] h**}**	9B [*n*[3B *n*]...] 68	Standard Set mode (See modes table) (default=none)
{CSI ? [*n*[;*n*]...] h**}**	9B 3F [*n*[3B *n*]...] 68	Private Set mode (See modes table) (default=none)
{CSI [*n*[;*n*]...] l**}**	9B [*n*[3B *n*]...] 6C	Standard Reset mode (See modes table) (default=none)

continued...

© 2005, QNX Software Systems

ASCII	Hex	Action
{CSI ? [*n*[;*n*]...] l**}**	9B 3F [*n*[3B *n*]...] 6C	Private Reset mode (See modes table) (default=none)
{CSI [*n*[;*n*]...] m**}**	9B [*n*[3B *n*]...] 6D	Select Graphic Rendition (See below) (default = 0)
{CSI *n* n**}**	9B *n* 6E	Device status 5=status 6=cursor/pos
{CSI [*r*[;*c*]] r**}**	9B [*r* [3B *c*]] 72	Set scroll region and home cursor
{CSI r**}**	9B 72	Disable scroll region & home cursor
{CSI s**}**	9B 73	Save cursor
{CSI u**}**	9B 75	Restore cursor
{CSI ! p**}**	9B 21 70	Soft reset
{CSI [*n*[;*n*]]]**}**	9B [*n* [3B *n*]...] 5D	Set default 1=underline 2=half-intensity 8=colorset (default=none)
{CSI = [*f* [;*d*]] B**}**	9B 3D [*f* [3B *d*]] 46	Set frequency(Hz) and duration (ms) for bell (default=100Hz, 250ms)
{CSI = [*n*] F**}**	9B 3D [*n*] 46	Set and Save foreground color

continued...

ASCII	Hex	Action
{CSI = [*n*] G**}**	9B 3D [*n*] 47	Set and Save background color

Graphic rendition

Number	Meaning
0	All attributes off (except charset (10, 11, 12))
1	Bold
2	Half intensity (default to cyan on color screen)
4	Underline (default to red on color screen)
5	Blink
7	Reverse
9	Invisible
10	Exit alternate char set (GR & GL are restored)
11	Enter PC-lower char set (GR & GL are ASCII; C0 & C1 are PC_LO except for ESC)
12	Enter PC-higher char set (GR, C1 & GL, C0 are PC_HI except for ESC)
21	Normal intensity (un-Bold)
22	Normal intensity (un-Half intensity)
24	Disable underline
25	Disable blink
27	Disable reverse
29	Visible

continued...

© 2005, QNX Software Systems

Number	Meaning
30-37	Set foreground color (30+*color_number*, see below)
39	set foreground to saved
40-47	Set background color (40+*color_number*, see below)
49	set background to saved

Color Numbers

color_number	Description
0	Black
1	Red
2	Green
3	Brown
4	Blue
5	Violet
6	Cyan
7	White

Modes

Mode string	Description
?1h	cursor key = Application
?1l	cursor key = ANSI (default)
?3h	132 column (Not Implemented)

continued...

Mode string	Description
?3l	80 column (default)
?5h	Reverse screen
?5l	Not Reverse screen (default)
?6h	Origin mode
?6l	Absolute mode
?7h	Auto wrap on
?7l	Auto wrap off (default)
?25h	Visible cursor (default)
?25l	Invisible cursor
?45h	Reverse wrap-around mode
?45l	No reverse wrap-around
?66h	keypad = Application
?66l	keypad = ANSI
?67h	Backspace key generates BS
?67l	Backspace key generates DEL

Switching protocols/terminal mode control sequence

ASCII	Hex	Action
{CSI ? [*p* [;*m*]] + q}	9B 3F *p* 3B *m* 2B 71	Set protocol and reset mode. Default *p*=0; *m*=0

© 2005, QNX Software Systems

Where	Means
p = 0	QNX4 emulation
p = 1	ANSI emulation
m = 0	No change
m = 1	Standard QNX defaults
m = 2	G2 is set to PC_HI & Backspace generates 08
m = 3	Wrap-around off
m = 4	G2 is set to PC_HI & reverse wrap-around enabled

Returning to ANSI mode from QNX4 mode

ASCII	Hex	Action
ESC ? 1 + q	1B 3F 31 2B 71	set to ANSI mode from QNX4 mode

Mapping from QNX keyboard to ANSI keys

Key	Normal	Shift	Ctrl	Alt
Enter	**CR**	**CR**	**CR**	**CR**
Tab	**TAB**	**CSI Z**	**CSI z**	
BS	**BS**	**RUB**	**RUB**	**BS**
ESC	**ESC**	**ESC**	**ESC**	**ESC**
F1	**SS3 P**	**SS3 p**	**CSI 1~**	**CSI 17~**
F2	**SS3 Q**	**SS3 q**	**CSI 2~**	**CSI 18~**
F3	**SS3 R**	**SS3 r**	**CSI 3~**	**CSI 19~**

continued...

Key	Normal	Shift	Ctrl	Alt
F4	SS3 S	SS3 s	CSI 4~	CSI 20~
F5	SS3 T	SS3 t	CSI 5~	CSI 21~
F6	SS3 U	SS3 u	CSI 6~	CSI 22~
F7	SS3 V	SS3 v	CSI 7~	CSI 23~
F8	SS3 W	SS3 w	CSI 8~	CSI 24~
F9	SS3 X	SS3 x	CSI 9~	CSI 25~
F10	SS3 Y	SS3 y	CSI 10~	CSI 26~
F11	SS3 Z	SS3 z	CSI 11~	CSI 27~
F12	SS3 A	SS3 a	CSI 12~	CSI 28~
Home	CSI H		CSI h	CSI H
↑	CSI A		CSI a	CSI A
PgUp	CSI V		CSI v	CSI V
Minus	CSI S		CSI s	CSI S
←	CSI D		CSI d	CSI D
kpd 5	CSI G		CSI g	CSI G
→	CSI C		CSI c	CSI C
Plus	CSI T		CSI t	CSI T
End	CSI Y		CSI y	CSI Y
↓	CSI B		CSI b	CSI B
PgDn	CSI U		CSI u	CSI U
Ins	CSI @		CSI ‘	CSI @
Del	CSI P		CSI p	CSI P
Prt	NOP	NOP	NOP	NOP

continued...

© 2005, QNX Software Systems

Key	Normal	Shift	Ctrl	Alt
SysRq	**NOP**	**NOP**	**NOP**	**NOP**
a	**a**	**A**	**SOH**	**SS2 a**
b	**b**	**B**	**STX**	**SS2 b**
c	**c**	**C**	**ETX**	**SS2 c**
d	**d**	**D**	**EOT**	**SS2 d**
e	**e**	**E**	**ENQ**	**SS2 e**
f	**f**	**F**	**ACK**	**SS2 f**
g	**g**	**G**	**BEL**	**SS2 g**
h	**h**	**H**	**BS**	**SS2 h**
i	**i**	**I**	**HT**	**SS2 i**
j	**j**	**J**	**LF**	**SS2 j**
k	**k**	**K**	**VT**	**SS2 k**
l	**l**	**L**	**FF**	**SS2 l**
m	**m**	**M**	**CR**	**SS2 m**
n	**n**	**N**	**SO**	**SS2 n**
o	**o**	**O**	**SI**	**SS2 o**
p	**p**	**P**	**DLE**	**SS2 p**
q	**q**	**Q**	**DC1**	**SS2 q**
r	**r**	**R**	**DC2**	**SS2 r**
s	**s**	**S**	**DC3**	**SS2 s**
t	**t**	**T**	**DC4**	**SS2 t**
u	**u**	**U**	**NAK**	**SS2 u**
v	**v**	**V**	**SYN**	**SS2 v**

continued...

Key	Normal	Shift	Ctrl	Alt
w	**w**	**W**	**ETB**	**SS2 w**
x	**x**	**X**	**CAN**	**SS2 x**
y	**y**	**Y**	**EM**	**SS2 y**
z	**z**	**Z**	**SUB**	**SS2 z**

Examples:

Start **Dev.ansi**; use defaults:

```
Dev.ansi  &
```

Allow 4 virtual consoles:

```
Dev.ansi -n 4 &
```

Files:

Dev.ansi closes its standard input, standard output and standard error immediately upon startup.

Dev.ansi will cause **Dev** to adopt a number of character special files under its directory (**/dev**). Usually these will be named **ansi***n*, where *n* will range from 1 to the number of console devices specified by the **Dev.ansi -n** option. The base filename (**ansi**) will be different if the **-N** is specified with an alternate name.

Exit status:

Dev.ansi will terminate only upon receipt of a signal or upon encountering a problem during startup.

0 received a SIGTERM signal and performed a clean shutdown

5 Failed to initialize display

© 2005, QNX Software Systems

12 Failed to mount driver with **Dev**

19 Failed to arm driver with **Dev**

23 Failed to register driver with **Dev**

A common cause of failure on startup is that **Dev** has hit its maximum number of devices and rejects the driver's attempt to register. If this happens, increase the value of **Dev**'s **-n** option.

See also:

Dev, **Dev.***, **cfont**, **kbd**, **stty**, **tinit**

QNX *Installation & Configuration*

Dev.con

QNX Console driver (QNX)

Syntax:

Dev.con [*options***...] &**

Options:

-A Start in ANSI terminal mode. (See **Dev.ansi**.)

-C *number* The size of the canonical input buffer (default is 256 bytes).

-h Enable hardware scrolling.

-H Disable hardware scrolling. (Default)

-I *number* The size of the raw input buffer (default is 256 bytes).

-J Disable Japanese keyboard mapping.

-k *rate*[**,***delay*]

Specify keyboard autorepeat rate (in Hz) and, optionally, delay (in milliseconds). (default: 30Hz, 500ms)

-n *number* The maximum number of virtual consoles (default is 1).

-N *name* Use this prefix for the device name. If a full path (i.e. one which includes the directory under which the device should be registered) is not specified, the name will implicitly be registered under **/dev**. (default is **/dev/con**)

-O *number* The size of the output buffer (default is 2048 bytes).

-Q Start in QNX terminal emulation mode. (default)

© 2005, QNX Software Systems

Description:

The **Dev.con** driver manages *console devices* in a QNX system.

Remember that **Dev.con** can be started only after the Device Manager (**Dev**) has been started.

Dev.con is a link to **Dev.ansi**. The two drivers have identical capabilities, but differ in their default terminal emulation modes. (**Dev.con** = **Dev.ansi -Q** and **Dev.ansi** = **Dev.con -A**). See **Dev.ansi** for information about changing terminal emulation modes on-the-fly on a particular console device.

Control codes

ASCII	Hex	Description
LF	0A	move the cursor down one line (if the cursor is positioned on the last line of the screen, the screen scrolls up one line)
CR	0D	move the cursor to the first column of the current row
BS	08	enter a nondestructive backspace (wraps to the end of the previous line when at the beginning of the line; wraps to the bottom right corner of the screen when at the top left corner of screen)
FF	0C	clear the entire screen to spaces (uses the fill color specified with **ESC !** and moves the cursor to the top left corner)
BEL	07	generate a short tone on the system console or ring the bell on a terminal

Escape sequences

ASCII	Hex	Description
ESC (	1B 28	turn on inverse video mode (if a terminal supports color, this also swaps the foreground and background colors set by the **ESC @** sequence)
ESC)	1B 29	turn off inverse video
ESC [	1B 5B	turn on underline mode (PC monochrome text adapters only)
ESC]	1B 5D	turn off underline mode
ESC **{**	1B 7B	turn on blinking mode
ESC **}**	1B 7D	turn off blinking mode
ESC **<**	1B 3C	turn on high-intensity mode
ESC **>**	1B 3E	turn off high-intensity mode
ESC y *c*	1B 79 (30-32)	define cursor type to be *c*, where ASCII **0** (Hex 30) is no cursor, **1** (Hex 31) is a normal cursor, and **2** (Hex 32) is a block cursor.
ESC = *r c*	1B 3D (20-39) (20-70)	position the cursor at the screen coordinates *r* and *c*, where *r* is the row and *c* is the column; both *r* and *c* are offset by 20 hex (a space).

continued...

© 2005, QNX Software Systems

ASCII	Hex	Description
ESC Y*r c*	1B 59 (20-39) (20-70)	synonymous to ESC =
ESC A	1B 41	move cursor up one line without wrap or scrolling
ESC B	1B 42	move cursor down one line without wrap or scrolling
ESC C	1B 43	move cursor right one column without wrap
ESC D	1B 44	move cursor left one column without wrap
ESC H	1B 48	move cursor to the top left corner of the screen
ESC h	1B 68	turn auto line wrapping on
ESC i	1B 69	turn auto line wrapping off
ESC a	1B 61	move cursor up with wrap
ESC b	1B 62	move cursor down with wrap
ESC c	1B 63	move cursor right with wrap
ESC d	1B 64	move cursor left with wrap
ESC I	1B 49	reverse linefeed (the screen scolls down when the cursor is positioned on the top line of the screen)

continued...

ASCII	Hex	Description
ESC E	1B 45	insert line at cursor location
ESC F	1B 46	delete line at cursor location
ESC J	1B 4A	clear to end of screen with spaces (uses the fill color specified with the **ESC !** sequence on color terminals)
ESC K	1B 4B	clear to end of line with spaces (uses the fill color specified with **ESC !** on color terminals)
ESC e	1B 65	insert space at cursor position
ESC f	1B 66	delete character at cursor position
ESC g *n c*	1B 67 (20-ff) (0-ff)	repeat the character *c* the number of times specified by *n*, where *n* is offset by 20 hex

continued...

© 2005, QNX Software Systems

ASCII	Hex	Description
ESC S	1B 53	save color and attribute information (This overrides any previously saved information. The cursor position isn't saved. QNX programs, such as editors that change the colors, assume that a default has been set on entry and use **ESC R** to restore the default on exit.)
ESC R	1B 52	restore the color and attribute information saved with **ESC S** (the cursor position isn't affected)
ESC @ *fc bc*	1B 40 (30-37) (30-37)	define the foreground and background color of displayed characters, where *fc* is the color of the cursor, and *bc* is the fill color. Both *fc* and *bc* represent a single character ASCII digit (**0** thru **7**) taken from the color table, below.
ESC ! *fc bc*	1B 21 (30-37) (30-37)	set the fill color that is used both for blank lines created by scrolling and for the entire screen when clearing with a form feed.

Color codes

ASCII Character	Color
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	White

Examples:

Start **Dev.con**; use defaults:

```
Dev.con &
```

Allow 4 virtual consoles:

```
Dev.con -n 4 &
```

Files:

Dev.con closes its standard input, standard output and standard error immediately upon startup.

Dev.con will cause **Dev** to adopt a number of character special files under its directory (**/dev**). Usually these will be named **con***n*, where *n* will range from 1 to the number of console devices specified by the **Dev.con -n** option. The base filename (**con**) will be different if the **-N** is specified with an alternate name.

© 2005, QNX Software Systems

Exit status:

Dev.con will terminate only upon receipt of a signal or upon encountering a problem during startup.

0	received a SIGTERM signal and performed a clean shutdown
5	Failed to initialize display
12	Failed to mount driver with **Dev**
19	Failed to arm driver with **Dev**
23	Failed to register driver with **Dev**

A common cause of failure on startup is that **Dev** has hit its maximum number of devices and rejects the driver's attempt to register. If this happens, increase the value of **Dev**'s **-n** option.

See also:

Dev, **Dev.ansi**, **Dev.***, **cfont**, **kbd**, **stty**, **tinit**

QNX *Installation & Configuration*

Syntax:

Dev.par [*options***] &**

Options:

-b *number*	Use the printer number, as defined by the BIOS, where *number* can be **1**, **2**, **3**, or **4**.
-c	Check *open()*s. If the printer isn't online and READY, any *open()*s are refused.
-e	Abort output if the printer is in an error condition.
-f	Send data faster than the IEEE parport specifications.
-N *name*	Use this prefix for the device name. If a full path (i.e. one which includes the directory under which the device should be registered) is not specified, the name will implicitly be registered under **/dev**. (default is **/dev/par**)
-O *number*	The size of the output buffer (default is 1000 bytes).
-P *priority*	The priority to run at when printing (default: 9)
-p *number*	The I/O address (in hex) of the parallel port (required only when using a non-standard parallel port that the BIOS is unable to find).
-s *number*	Don't wait forever on the printer. Instead, wait at most this many seconds before discarding data.
-w *number*	Busy-wait on the printer this many microseconds before reverting to periodic polling (default is 1000 microseconds or 1 millisecond).

Description:

The **Dev.par** driver manages *parallel* devices in a QNX system.

A unique instance of **Dev.par** must be started for each parallel device to be supported. Each device must have a unique name, so the **-N** *name* option must be specified to each additional driver.

If **Dev.par** is started without specifying any I/O ports, it uses the first parallel port found by the system BIOS (**LPT1**). If there aren't any parallel ports, **Dev.par** terminates.

Normally, **Dev.par** will busy-wait on a printer at low priority, waiting for the printer to accept new characters. This behavior results in maximum throughput to the printer without significantly impeding overall system response. Nevertheless, it also means that **Dev.par** will wait forever if the printer ever goes offline (e.g. paper out) until the offline condition is removed (e.g. paper added). Therefore, it is sometimes worthwhile to put a limit on the length of time **Dev.par** should wait for the printer to go online. To set this timeout, use the **-s***number* option. If the printer stays offline the specified number of seconds, any pending data is simply thrown away. This is useful in unattended environments where it's preferable to throw away logging output rather than block the logging processes themselves when the output buffers fill up.

The priority at which **Dev.par** polls the parallel port is 9 by default. To lower (or raise) this, use the **-P** *priority* option to specify a different priority.

Dev.par must be run as **root** and may not be started until the Device Manager (**Dev**) has been started.

Examples:

Support **LPT1** as **/dev/par1**:

```
Dev.par &
```

Support **LPT2** as **/dev/prn1**:

```
Dev.par -b 2 -N prn &
```

Support port at 278 hex as **/dev/laser1**

```
Dev.par -p 278  -N laser &
```

Don't wait on the printer on **/dev/par**. If the printer remains unavailable for longer than 30 seconds, flush all data:

```
Dev.par -s 30 &
```

Files:

Dev.par closes its standard input, standard output and standard error immediately upon startup.

Dev.par will cause **Dev** to adopt a single character special file under its directory (**/dev**). Usually this file will be named **/dev/par1**, but will be different if the **-N** is specified with an alternate name.

Exit status:

Dev.par will terminate only upon receipt of a signal or upon encountering a problem during startup.

0	received a SIGTERM signal and performed a clean shutdown *or* no **-p** port was specified and **Dev.par** failed to detect any parallel ports on the machine.
12	Failed to mount driver with **Dev**
19	Failed to arm driver with **Dev**
23	Failed to register driver with **Dev**

© 2005, QNX Software Systems

A common cause of failure on startup is that **Dev** has hit its maximum number of devices and rejects the driver's attempt to register. If this happens, increase the value of **Dev**'s **-n** option.

See also:

Dev, **Dev.***, **stty**

QNX *Installation & Configuration*

Dev.pty

Pseudo-tty driver (QNX)

Syntax:

Dev.pty [-C *number***] [-I** *number***] [-O** *number***]**
[-l *letter***] [-n** *number***] [-N** *mastername***[,**_slavename_**] &**

Options:

-C *number* The size of the canonical input buffer (default is 256).

-I *number* The size of the raw input buffer (default is 512).

-l *letter* ("el") The last letter of the device name (default is **p**, which results in **/dev/ttyp**). Other choices are **q**, **r**, or **s**.

-n *number* The maximum number of pseudo tty pairs (default is 4). Device pairs will be numbered incrementally in hexadecimal notation. Values greater than 16 (**f**) are possible but are not recommended for UNIX compatibility reasons.

-N *mastername*[**,***slavename*]

The name to use for the master (and optionally, slave) sides of the pseudo tty devices (default is **/dev/pty,/dev/tty**). If full paths (i.e. ones that include the directory under which the devices should be registered) are not specified, the names will implicitly be registered under **/dev**.

-O *number* The size of the output buffer (default is 512).

Description:

The **Dev.pty** driver manages pseudo tty devices (called *ptys*) in a QNX system. Pseudo devices are allocated in pairs; the members of each pair are internally connected to each other. Any data written to the master side (e.g. **/dev/ptyp0**) will be available for reading on the slave side (e.g. **/dev/ttyp0**).

Typically, a program will open the master side of a pty for read/write and then launch another program (e.g. a Shell) on the corresponding slave device. The slave program will think it's running on a normal tty device. Nevertheless, all the data that the slave writes will go to the master and all the data that the slave reads will really be data generated by the master.

Remember that **Dev.pty** must be run as **root** and cannot be started until the Device Manager (**Dev**) has been started.

QNX programs that open ptys (e.g. **telnetd**, **rlogind**, etc.) search an extended set of ptys. The search order is:

- **/dev/pty**[**pqrs**][**0-f**]
- **/dev/pty/0/p**[**0-ff**]
- **/dev/pty/1/p**[**0-ff**]
- **/dev/pty/2/p**[**0-ff**]

  ⋮
- **/dev/pty/999/p**[**0-ff**]

The search stops upon encountering a non-existent blank directory. If the directory **/dev/pty/1** doesn't exist, the search will stop there rather than go on to search **/dev/pty/2** and so on.

Applications that wish to use the same pty search ordering can open ptys using the *open_pty()* function in the **unix** library:

```
int open_pty(int *masterfd, int *slavefd,
             char *name, struct termios *termp,
             struct winsize *winp);
```

(The arguments *name*, *termp* and *winp* may be NULL.)

Examples:

Start **Dev.pty**; use defaults (this would create **/dev/ptyp0** through **/dev/ptyp3** and **/dev/ttyp0** through **/dev/ttyp3**):

```
Dev.pty &
```

Create 16 device pairs called **/dev/ptyq0** through **/dev/ptyqf** and **/dev/ttyq0** through **/dev/ttyqf**:

```
Dev.pty -n 16 -l q &
```

Create 32 device pairs under a directory **/dev/pseudo** that is managed by a second **Dev** administrator. Name the devices **/dev/pseudo/m0** through **/dev/pseudo/m1f** and **/dev/pseudo/q0** through **/dev/pseudo/q1f**:

```
Dev -N /dev/pseudo -n65 &
Dev.pty -n32 -N /dev/pseudo/m,/dev/pseudo/s &
```

Files:

Dev.pty closes its standard input, standard output and standard error immediately after startup initializations have been completed. An error message is written to standard error if an initialization problem is encountered.

Dev.pty will cause **Dev** to adopt a number of pairs of character special files under its directory (**/dev**). Usually pairs will be named **pty***cn*/**tty***cn* where *c* is **p** (the default) or the letter specified by the **-l** option, and *n* will range from 1 to the number of pseudo-tty devices specified by the **Dev.pty -n** option.

Exit status:

Dev.pty will terminate only upon receiving a signal, encountering a problem during startup, or having to *fork()* due to not being started in the "background" (i.e. the ampersand [**&**] was missing on its command-line invocation).

© 2005, QNX Software Systems

0 received a signal and performed a clean shutdown *OR* not started in the background from the shell and successfully forked to another process.

>0 An error occurred during startup. An error message was written to standard error.

A common cause of failure on startup is that **Dev** has hit its maximum number of devices and rejects **Dev.pty**'s attempt to register. If this happens, increase the value of **Dev**'s **-n** option.

See also:

Dev, **Dev.***, **stty**, **tinit**

Syntax:

Dev.ser [[*options***]... [***portspec***...]]... &**

Options:

The **Dev.ser** driver does not follow the utility syntax conventions. All options do not necessarily appear before operands; they may be intermixed. Options apply to those operands which follow them on the command line.

portspec One of:

port[**,***irq*[**,p**]] A serial I/O port (in hex). A "**,***irq*" after *port* specifies the IRQ (in decimal) used by the port. By default the interrupt will be autodetected. A "**,p**" after *port* and *irq* tells **Dev.ser** that the device is going to be found on a PCMCIA card (any socket).

p[*socket*[**,***socket*]...] A PCMCIA serial port in one of the socket(s) named.

-b *number* The initial baud rate for subsequent ports (default is 9600).

-C *number* The size of the canonical input buffer (default is 256 bytes).

-d Set initial state of DSR to on (default).

-D Set initial state of DSR to off.

-e Set all subsequent ports to "edited" mode.

-E Set all subsequent ports to "raw" mode (default).

© 2005, QNX Software Systems

-f	Enable hardware flow control on all subsequent ports and lock it on (default).
-F	Disable locked hardware flow control on all subsequent ports.
-h *number*	The flow control high-water mark (as a number of bytes of the input buffer in use) for subsequent ports (default is 75% of the input buffer — see **-I**, below).
-I *number*	The size of the raw input buffer (default is 2048 bytes).
-l *number*	("el") The flow control low-water mark (as a number of bytes of the input buffer in use) for subsequent ports (default is 25% of input buffer — see **-I**, above).
-m	Disable modem carrier processing (HUP).
-M	Enable modem carrier processing (HUP). Default.
-N *name*	Use this prefix for the device name. If a full path (i.e. one which includes the directory under which the device should be registered) is not specified, the name will implicitly be registered under **/dev**. (default is **/dev/ser**)
-O *number*	The size of the raw output buffer (default is 2048 bytes).
-p	Enable packet mode. (Modem status change causes "packet" event.)
-P	Disable packet mode (default)
-r	Set initial state of **RTS** to on (default).
-R	Set initial state of **RTS** to off.
-s	Enable split of output flow control signals — transmission requires **CTS** only (default).

-S	Disable split of output flow control signals — transmission requires both **CTS** and **DSR**.
-t *number*	Set 16550 FIFO threshold to 1, 4, 8 or 14 to help minimize CPU utilization. 0 disables the FIFO. (Default 0)
-w	Enable hardware shutdown on SIGPWR.
-W	Disable hardware shutdown on SIGPWR (default)
-x	Enable software flow control (**XON**/**XOFF**) on all subsequent ports and lock it on.
-X	Disable locked software flow control on all subsequent ports (default).
-y	Set baudrate divisor (default is 115200).

Description:

The **Dev.ser** driver manages standard PC/AT and PCMCIA serial port devices in a QNX system, including multiport devices which appear as standard serial ports on the bus but which may share a common interrupt. Serial "smartcards" require special drivers from other companies (such as the smartcard manufacturer).

If no serial ports are specified on the command line, **Dev.ser** scans the hardware, looking for a port at 3F8 (**COM1**) or a port at 2F8 (**COM2**), or both. If **COM1** is present, **Dev.ser** assumes that it uses interrupt **IRQ 4**; if **COM2** is present, **Dev.ser** assumes that it uses interrupt **IRQ 3**. If the scan shows that neither of **COM1** nor **COM2** is present, **Dev.ser** terminates.

Locking flow control prevents application programs — with the exception of **stty** — from changing the flow control modes.

When using **-t** to enable the 16550 FIFO, be aware that it will not generate a receive interrupt until either the FIFO is full or 4 idle character times at the current baud rate have elapsed. This could have ramifications for low-latency protocol applications. Note that the

16550 will always buffer incoming data. The **-t** option affects only the generation of interrupts.

Dev.ser must be run as **root** and may be started only after the Device Manager (**Dev**) has been started.

Examples:

Scan for **COM1** and **COM2**:

```
Dev.ser &
```

Set port at 3F8 as a raw device at 1200 baud, and port at 2F8 as an edited device at 9600 baud:

```
Dev.ser  -E -b 1200 3f8,4  -e -b 9600 2f8,3  &
```

Specify a standard **COM1** port at 3F8 as well as four ports on a multiport card:

```
Dev.ser  3f8,4 280,3 288,3 290,3 298,3  &
```

Do the same, but allow **Dev.ser** to autodetect the IRQs used for the four ports:

```
Dev.ser  3f8 280 288 290 298 &
```

Set the driver up for one serial port **/dev/ser1** which will be a PCMCIA card in socket 1 or 2:

```
Dev.ser p1,2 &
```

Set the driver up for two serial ports (**/dev/ser1** and **/dev/ser2** one of which will be a PCMCIA card in socket 1 and the other a PCMCIA card in socket 2:

```
Dev.ser p1 p2 &
```

Set the driver up for two serial ports at 0x3f8/irq3 and 0x2f8/irq4, plus a third serial port at 0x3e8/irq7 which will be found on a PCMCIA card which may be in any socket:

```
Dev.ser 3f8,3 2f8,4 3e8,7,p &
```

Files:

Dev.ser closes its standard input, standard output and standard error immediately upon startup.

Dev.ser will cause **Dev** to adopt a number of character special files under its directory (**/dev**). These will usually be named **ser***n*, where *n* will range from 1 to the number of serial devices encountered during the scan of possible port locations or specified explicitly on the command line. The base filename (**ser**) will be different if the **-N** is specified with an alternate name.

Exit status:

Dev.ser will terminate only upon receipt of a signal or upon encountering a problem during startup.

0	Received a signal and performed a clean shutdown *or* **Dev.ser** scanned for serial ports (none specified on the command line) and found none.
12	Failed to mount driver with **Dev**
19	Failed to arm driver with **Dev**
23	Failed to register driver with **Dev**
34	Too many ports specified on the command line. (**>**64)

© 2005, QNX Software Systems

A common cause of failure on startup is that **Dev** has hit its maximum number of devices and rejects the driver's attempt to register. If this happens, increase the value of **Dev**'s **-n** option.

See also:

Dev, **Dev.***, **modem**, **Mouse**, **Net.fd**, **qtalk**, **qcp**, **stty**, **tinit**

QNX *Installation & Configuration*

Syntax:

Report disk space free:

df [-abdhw] [-n *node***] [***file***...]**

Report mapping points:

df -m [-w] [-n *node***]**

Options:

-a (QNX extension) Display all block devices, with the exception of any for which no size information is available. Information will be displayed for a block device even if it's not mounted as a filesystem.

-b (QNX extension) Display disk space in blocks of 512 bytes (default is units of 1K).

-d (QNX extension) Show duplicate devices (i.e. multiple prefixes or mounts to the same device). If **-d** isn't specified, only the first mapping that resolves to a particular filesystem will be shown.

-h (QNX extension) Display a header.

-m (QNX extension) Instead of displaying free disk space, display a map of prefix mappings available under **/** on the specified node (or the node **df** is running on if **-n** isn't specified). This will show the name the directory appears as under **/**, and the actual directory it maps to (including its node prefix ***//n/***). This option is very useful when debugging prefix problems in a network.

-n *node* (QNX extension) Use the specified node's prefix tree to resolve the possible pathname(s) to mounted filesystems.

-w (QNX extension) Warn about prefix mappings that hit or exceed five levels of nesting. This usually indicates that a namespace hasn't been "sensibly" configured.

file The pathname of a file within the hierarchy of the desired filesystem.

Description:

The **df** utility provides disk space information as well as mappings of prefix namespace resolution under **/**. The utility has two invocation modes.

df [**-abdhw**] [**-n** *node*] [*file*...]

In this mode, **df** prints the amount of available space for filesystems to which the user has read access. Filesystems are specified by *file* operands. When no *file* operands are specified, **df** produces information for every filesystem available under **/** (i.e. nothing that requires a **//***n***/** prefix).

df -m [**-w**] [**-n** *node*]

In this (map) mode, **df** prints prefix mapping points (directories) available under **/** and the actual filesystem point they resolve to. The mapping point is displayed as a full path, with a node prefix. You invoke this mode by specifying the **-m** option.

When displaying disk space information, **df** produces output that consists of one line of information for each specified filesystem. If **-h** is specified, the output contains a one-line header indicating the meaning of each column. The fields on the output lines are, in order:

filesystem name

The name of the filesystem (i.e. the name of the block special file, including node number).

total space The total formatted capacity of the disk drive or partition in question.

total available user space

The total amount of space available to the user. This is the sum of *space used* + *space free*.

space used The total amount of space allocated to all files within the filesystem.

space free The total amount of space available within the filesystem for the creation of new files by unprivileged users. When this figure is less than or equal to zero, it's not possible to create any new files on the filesystem without first deleting others.

percentage used

The percentage of the normally available space that is currently allocated to files. This is the fraction:

space used÷*total available user space*

expressed as a percentage, rounded to the nearest integer. If *total available user space* is zero, then this percentage is displayed as "**100%**". This percentage may be greater than 100 if *total space* isn't equal to *total available user space*.

filesystem mount point

The directory below which the filesystem hierarchy appears.

If a *file* operand is specified, this column will show the actual mount point of the filesystem within which the *file* resides, including the node prefix.

If no *file* operand is specified, this column will show the path through which the listed filesystem is accessed. You should note that depending on prefix

mappings, a pathname may not be a mount point but rather an alias prefix.

Note that entries will be skipped when information on the filesystem cannot be obtained, except when the **-a** option is used, in which case block special files and their sizes will be listed even if they are not mounted as filesystems.

Examples:

Print information about the filesystems accessible under **/**:

```
df
```

Print information about the filesystem that **/usr/src** resides on:

```
df /usr/src
```

Show information on prefix mappings accessible under **/**, with warnings about deep-nesting:

```
df -mw
```

Files:

df writes its output to the standard output. If an error occurs, a diagnostic message will be written to the standard error. The standard input is not used.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

du, **prefix**

diff

© 2005, QNX Software Systems

Report the difference between two files (POSIX)

Syntax:

diff [-C *n***|-c|-e|-D** *string***|-s|-n] [-L** *label1* **[***label2***]]**
[-br] *file1* *file2*

Options:

-b Ignore trailing **blank** characters and treat other strings of **blank** characters as equivalent.

-c Produce output in a form that provides three lines of context (see "Context output format").

-C *n* Produce output in a form that provides *n* lines of context (see "Context output format").

-D *string* (QNX extension) Produce output merging *file1* and *file2* using C Language preprocessor controls (see "C output format").

-e Produce output in a form suitable as input for the Unix line editor. The editor can then be used to convert *file1* into *file2* (see "Edit output format").

-L *label1* [*label2*]

(historical extension) Print these labels for context information (used by RCS).

-n Produce output as RCS-style differences.

-r When *file1* and *file2* are both directories, apply **diff** recursively to files and directories of the same name (see "Directory output format").

-s (QNX extension) Produce only a summary of the differences (see "Summary output format").

file1 The pathname of a file to be compared.

file2 The pathname of a file to be compared.

Description:

The **diff** utility reports the differences between two files. For any two files, there may be several correct interpretations of the differences between them. The **diff** utility attempts to generate the smallest number of additions, changes, and deletions required to convert *file1* into *file2*. No output is produced if the files are identical.

The **diff** utility may generate one of the following output formats:

- default output format
- edit output format
- context output format
- directory output format
- C output format
- summary output format

Default output format

The default output contains lines of this form:

range1 cmd range2

where *cmd* is one of: **a** (add), **c** (change), or **d** (delete).

The options *range1* and *range2* are the line numbers that differ between *file1* and *file2* respectively.

A range will consist of "*startline*" or "*startline*,*endline*" if more than one line is affected.

Each of the default output lines is followed by a sequence of zero or more affected lines from *file1*, preceded by the **<** character, then a sequence of zero or more affected lines from *file2* preceded by the **>** character. If lines are actually printed from both (i.e. a "change" command), then a line of three dashes (**---**) is output between the listing of lines from *file1* and *file2*.

Edit output format

The edit output is similar to the default output, except only the edit commands necessary to convert *file1* into *file2* are output with the relevant text.

Context output format

The context output consists of an identification line for each file, containing the filename and modification date of the file. If the **-L** option is specified, labels will also be printed. Each of the remaining output lines is prefixed by one of the following characters:

Character:	Meaning:
space	an unaffected line, shown for context
-	a line to be deleted from the first file
+	a line to be added to the second file
!	a line to be changed

Directory output format

For each file present in both directories, if the files are not identical the string:

"**diff** *diff_options filename1 filename2*"

is produced. The field *diff_options* refers to the options given to **diff** to check the two files. Each file that is present in only one directory is reported by the string:

"**Only in** *directory_name*:*filename*"

Subdirectories that are common to the two directories are reported as:

"**Common subdirectories:** *pathname1* **and** *pathname2*"

C output format

The C output format is a merged version of *file1* and *file2* with C preprocessor controls included so that compiling without defining *string* is equivalent to compiling *file1*. Defining *string* makes it equivalent to compiling *file2*.

Summary output format

The summary output format is one line that contains two decimal numbers separated by a single **tab** character. The numbers represent the number of lines inserted and deleted to convert *file1* into *file2*.

Examples:

The following examples show how the **diff** utility operates on two simple files containing seven lines each:

The first file, **myfile**, contains

```
a
b
c
d
e
f
g
```

and the second file, **yourfile**, contains:

```
w
a
b
x
y
z
e
```

Default output example

Notice that to change **myfile** into **yourfile**, the following could be performed: insert a **w** before line 1, change lines 3 and 4 from **c d** to **x y z**, delete lines 6 and 7, which were **f** and **g**, respectively.

Invoking **diff myfile yourfile** gives the following Default output:

```
1a1
> w
3,4c4,6
< c
< d
---
> x
> y
> z
6,7d7
< f
< g
```

Edit output example

Executing **diff -e myfile yourfile** gives the following Edit output:

```
0a
w
.
4,5c
x
y
z
.
8,9d
```

You may notice that the Edit output has different line numbering conventions from the Default output. This is because the script must "track" the changes it makes to the original since the Unix line editor

(**ed**) evaluates the line numbers based upon the current state of the file.

Files:

The input files processed by **diff** must be ASCII text files.

Exit status:

0	No differences were found.
1	Differences were found.
>1	An error occurred.

Caveats:

The **diff** utility doesn't always generate the minimum set of differences between two files. It does, however, always generate a set of differences that may be used by a program to derive one file from another.

When you use the **-D** option, **diff** simply inserts preprocessor directives without regard for existing directives, which it may contradict.

See also:

cksum, **cmp**, **comm**, **diff3**, **wc**

diff3

© 2005, QNX Software Systems

Show differences among three files (GNU)

Syntax:

diff3 [*options***...]** *mine older yours*

Options:

-a Treat all files as text and compare them line-by-line, even if they do not appear to be text.

-A Incorporate all changes from *older* to *yours* into *mine*, surrounding all conflicts with bracket lines. See section "Marking conflicts".

-e Generate an **ed** script that incorporates all the changes from *older* to *yours* into *mine*. See section "Selecting which changes to incorporate". (**ed** is not included with the QNX distribution)

-E Like **-e**, but output unmerged changes, bracketing overlaps. See section "Marking conflicts".

-i Generate **w** and **q** commands at the end of the **ed** script for System V compatibility. This option must be combined with one of the **-AeExX3** options, and may not be combined with **-m**. See section "Saving the changed file". (**ed** is not included with the QNX distribution)

-L *label* Use the label *label* for the brackets output by the **-A**, **-E** and **-X** options. This option may be given up to three times, one for each input file. The default labels are the names of the input files. Thus **diff3 -L X -L Y -L Z -m A B C** acts like **diff3 -m A B C**, except that the output looks like it came from files named **X**, **Y** and **Z** rather than from files named **A**, **B** and **C**. See section "Marking conflicts".

-m Apply the edit script to the first file and send the result to standard output. Unlike piping the output from **diff3** to **ed**, this works even for binary files and incomplete lines.

-A is assumed if no edit script option is specified. See section “Generating the merged output directly”.

-T Output a tab rather than two spaces before the text of a line in normal format. This causes the alignment of tabs in the line to look normal.

-v Output the version number of **diff3**.

-x Like **-e**, except output only the overlapping changes. See section “Selecting which changes to incorporate”.

-X Like **-E**, except output only the overlapping changes. In other words, like **-x**, except bracket changes as in **-E**. See section “Marking conflicts”.

-3 Like **-e**, except output only the nonoverlapping changes. See section “Selecting which changes to incorporate”.

mine older yours

These are the pathnames of the files to be compared. At most one of these three file names may be a dash (**-**), which tells **diff3** to read the standard input for that file.

Description:

The **diff3** utility is used to compare three files and show any differences among them. (**diff3** can also merge files; see section “Merging from a common ancestor”).

The normal **diff3** output format shows each hunk of differences without surrounding context. Hunks are labeled depending on whether they are two-way or three-way, and lines are annotated by their location in the input files.

Sample input files

Here are three sample files that we will use in numerous examples to illustrate the output of **diff3** and how various options can change it.

Sample file #1 — lao

This is the file **lao**:

```
The Way that can be told of is not the eternal Way;
The name that can be named is not the eternal name.
The Nameless is the origin of Heaven and Earth;
The Named is the mother of all things.
Therefore let there always be non-being,
  so we may see their subtlety,
And let there always be being,
  so we may see their outcome.
The two are the same,
But after they are produced,
  they have different names.
```

Sample file #2 — tzu

This is the file **tzu**:

```
The Nameless is the origin of Heaven and Earth;
The named is the mother of all things.

Therefore let there always be non-being,
  so we may see their subtlety,
And let there always be being,
  so we may see their outcome.
The two are the same,
But after they are produced,
  they have different names.
They both may be called deep and profound.
Deeper and more profound,
The door of all subtleties!
```

Sample file #3 — tao

This is the third sample file, called **tao**:

```
The Way that can be told of is not the eternal Way;
The name that can be named is not the eternal name.
The Nameless is the origin of Heaven and Earth;
The named is the mother of all things.

Therefore let there always be non-being,
```

```
  so we may see their subtlety,
And let there always be being,
  so we may see their result.
The two are the same,
But after they are produced,
  they have different names.

  -- The Way of Lao-Tzu, tr. Wing-tsit Chan
```

Detailed description of **diff3** normal format

Each hunk begins with a line marked "**====**". Three-way hunks have plain "**====**" lines, and two-way hunks have "**1**", "**2**", or "**3**" appended to specify which of the three input files differ in that hunk. The hunks contain copies of two or three sets of input lines each preceded by one or two commands identifying where the lines came from.

Normally, two spaces precede each copy of an input line to distinguish it from the commands. But with the **-T** option, **diff3** uses a tab instead of two spaces; this lines up tabs correctly.

Commands take the following forms:

file:*l***a** This hunk appears after line *l* of file *file*, and contains no lines in that file. To edit this file to yield the other files, one must append hunk lines taken from the other files. For example, "**1:11a**" means that the hunk follows line 11 in the first file and contains no lines from that file.

file:*r***c** This hunk contains the lines in the range *r* of file *file*. The range *r* is a comma-separated pair of line numbers, or just one number if the range is a singleton. To edit this file to yield the other files, one must change the specified lines to be the lines taken from the other files. For example, "**2:11,13c**" means that the hunk contains lines 11 through 13 from the second file.

If the last line in a set of input lines is incomplete, it is distinguished on output from a full line by a following line that starts with "\".

diff3 hunks

Groups of lines that differ in two or three of the input files are called *diff3 hunks*, by analogy with **diff** hunks. If all three input files differ in a **diff3** hunk, the hunk is called a *three-way hunk*; if just two input files differ, it is a *two-way hunk*.

As with **diff**, several solutions are possible. When comparing the files **A**, **B**, and **C**, **diff3** normally finds **diff3** hunks by merging the two-way hunks output by the two commands **diff A B** and **diff A C**. This does not necessarily minimize the size of the output, but exceptions should be rare.

For example, suppose the file **F** contains the three lines "**a**", "**b**", "**f**", the file **G** contains the lines "**g**", "**b**", "**g**", and the file **H** contains the lines "**a**", "**b**", "**h**". In this case the command **diff3 F G H** might produce the following output:

```
====2
1:1c
3:1c
  a
2:1c
  g
====
1:3c
  f
2:3c
  g
3:3c
  h
```

because it found a two-way hunk containing "**a**" in the first and third files and "**g**" in the second file, then the single line "**b**" common to all three files, then a three-way hunk containing the last line of each file.

An example of diff3 normal format

Here is the output of the command **diff3 lao tzu tao** (see section "Sample input files", for the complete contents of the files). Notice that it shows only the lines that are different among the three files.

```
====2
1:1,2c
3:1,2c
  The Way that can be told of is not the eternal Way;
  The name that can be named is not the eternal name.
2:0a
====1
1:4c
  The Named is the mother of all things.
2:2,3c
3:4,5c
  The named is the mother of all things.

====3
1:8c
2:7c
    so we may see their outcome.
3:9c
    so we may see their result.
====
1:11a
2:11,13c
  They both may be called deep and profound.
  Deeper and more profound,
  The door of all subtleties!
3:13,14c

    -- The Way of Lao-Tzu, tr. Wing-tsit Chan
```

Merging from a common ancestor

When changes have been made to copies of the same file, **diff3** can produce a merged output that contains both sets of changes together with warnings about conflicts.

One might imagine programs with names like **diff4** and **diff5** to compare more than three files simultaneously, but in practice the need rarely arises. You can use **diff3** to merge three or more sets of changes to a file by merging two change sets at a time.

The **diff3** utility can incorporate changes from two modified versions into a common preceding version. This lets you merge the sets of changes represented by the two newer files. Specify the common ancestor version as the second argument and the two newer versions as the first and third arguments, like this:

diff3 *mine older yours*

You can remember the order of the arguments by noting that they are in alphabetical order.

You can think of this as subtracting *older* from *yours* and adding the result to *mine*, or as merging into *mine* the changes that would turn *older* into *yours*. This merging is well-defined as long as *mine* and *older* match in the neighborhood of each such change. This fails to be true when all three input files differ or when only *older* differs; we call this a *conflict*. When all three input files differ, we call the conflict an *overlap*.

The **diff3** utility gives you several ways to handle overlaps and conflicts. You can omit overlaps or conflicts, or select only overlaps, or mark conflicts with special "**<<<<<<<**" and "**>>>>>>>**" lines.

The **diff3** utility can output the merge results as an **ed** script that that can be applied to the first file to yield the merged output. However, it is usually better to have **diff3** generate the merged output directly; this bypasses some problems with **ed**. Note that **ed** is not included with the QNX distribution.

Selecting which changes to incorporate

You can select all unmerged changes from *older* to *yours* for merging into *mine* with the **-e** option. You can select only the nonoverlapping unmerged changes with **-3**, and you can select only the overlapping changes with **-x**.

The **-e**, **-3** and **-x** options select only *unmerged changes*, i.e. changes where *mine* and *yours* differ; they ignore changes from *older* to *yours* where *mine* and *yours* are identical, because they assume that such

changes have already been merged. If this assumption is not a safe one, you can use the **-A** option. (See section "Marking conflicts").

Here is the output of the command **diff3** with each of these three options (see section "Sample input files", for the complete contents of the files). Notice that **-e** outputs the union of the disjoint sets of changes output by **-3** and **-x**.

Output of **diff3 -e lao tzu tao**:

```
11a

  -- The Way of Lao-Tzu, tr. Wing-tsit Chan
.
8c
  so we may see their result.
.
```

Output of **diff3 -3 lao tzu tao**:

```
8c
  so we may see their result.
.
```

Output of **diff3 -x lao tzu tao**:

```
11a

  -- The Way of Lao-Tzu, tr. Wing-tsit Chan
.
```

Marking conflicts

The **diff3** utility can mark conflicts in the merged output by bracketing them with special marker lines. A conflict that comes from two files *A* and *B* is marked as follows:

<<<<<<< *A*
lines from *A*
=======
lines from *B*
>>>>>>> *B*

A conflict that comes from three files *A*, *B* and *C* is marked as follows:

```
<<<<<<< A
lines from A
||||||| B
lines from B
=======
lines from C
>>>>>>> C
```

The **-A** option acts like the **-e** option, except that it brackets conflicts, and it outputs all changes from *older* to *yours*, not just the unmerged changes. Thus, given the sample input files (see section "Sample input files"), **diff3 -A lao tzu tao** puts brackets around the conflict where only **tzu** differs:

```
11a
||||||| tzu
They both may be called deep and profound.
Deeper and more profound,
The door of all subtleties!
=======

-- The Way of Lao-Tzu, tr. Wing-tsit Chan
>>>>>>> tao
.
11a
<<<<<<< lao
.
8c
so we may see their result.
.
2a
>>>>>>> tao
.
0a
<<<<<<< tzu
=======
.
```

The **-E** option outputs less information than the **-A** option, because it outputs only unmerged changes, and it never outputs the contents of the second file. Thus the **-E** option acts like the **-e** option, except that it brackets the first and third files from three-way overlapping changes. Similarly, **-X** acts like **-x**, except it brackets all its (necessarily overlapping) changes. For example, for the three-way overlapping change above, the **-E** and **-X** options output the following:

```
11a
||||||| tzu
They both may be called deep and profound.
Deeper and more profound,
The door of all subtleties!
=======

-- The Way of Lao-Tzu, tr. Wing-tsit Chan
>>>>>>> tao
.
11a
<<<<<<< lao
.
8c
so we may see their result.
.
2a
>>>>>>> tao
.
0a
<<<<<<< tzu
=======
.
```

If you are comparing files that have meaningless or uninformative names, you can use the **-L** *label* or **--label=***label* option to show alternate names in the "**<<<<<<<**", "**|||||||**", and "**>>>>>>>**" brackets. This option can be given up to three times, once for each input file. Thus **diff3 -A -L X -L Y -L Z A B C** acts like **diff3 -A A B C**, except that the output looks like it came from files named **X**, **Y** and **Z** rather than from files named **A**, **B** and **C**.

Generating the merged output directly

With the **-m** option, **diff3** outputs the merged file directly. This is more efficient than using **ed** to generate it, and works even with non-text files that **ed** would reject. If you specify **-m** without an **ed** script option, **-A** (**--show-all**) is assumed. (Note that **ed** is not included with the QNX distribution.)

For example, the command **diff3 -m lao tzu tao** (see section "Sample input files" for a copy of the input files) would output the following:

```
<<<<<<< tzu
=======
The Way that can be told of is not the eternal Way;
The name that can be named is not the eternal name.
>>>>>>> tao
The Nameless is the origin of Heaven and Earth;
The Named is the mother of all things.
Therefore let there always be non-being,
  so we may see their subtlety,
And let there always be being,
  so we may see their result.
The two are the same,
But after they are produced,
  they have different names.
<<<<<<< lao
||||||| tzu
They both may be called deep and profound.
Deeper and more profound,
The door of all subtleties!
=======

  -- The Way of Lao-Tzu, tr. Wing-tsit Chan
>>>>>>> tao
```

How diff3 merges incomplete lines

With **-m**, incomplete lines are simply copied to the output as they are found; if the merged output ends in an conflict and one of the input files ends in an incomplete line, succeeding "**|||||||**", "**=======**", or "**>>>>>>>**" brackets appear somewhere other than the start of a line because they are appended to the incomplete line.

Without **-m**, if an **ed** script option is specified and an incomplete line is found, **diff3** generates a warning and acts as if a newline had been present.

Saving the changed file

Traditional Unix **diff3** generates an **ed** script without the trailing **w** and **q** commands that save the changes. System V **diff3** generates these extra commands. GNU **diff3** normally behaves like traditional Unix **diff3**, but with the **-i** option it behaves like System V **diff3** and appends the **w** and **q** commands.

The **-i** option requires one of the **ed** script options **-AeExX3**, and is incompatible with the merged output option **-m**.

Exit status:

0 **diff3** was successful.

1 Some conflicts were found.

>1 An error occurred.

Contributing author:

GNU

See also:

cmp, **comm**, **diff**, **rcsmerge**

dinit

© 2005, QNX Software Systems

Initialize a QNX filesystem (QNX)

Syntax:

dinit [-bhpqr] [-B *filename***|-O] [-L** *label***|-l** *label***]**
[-f *bootfile***] [-m** *message***]** *drive*

Options:

-b	Do not initialize the filesystem. Write OS loader (only) to disk.
-B *filename*	Use 512 byte OS loader from this file rather than the standard QNX OS loader. Note that on a partitioned hard disk, the OS loader is the second loader, not the primary bootstrap loader which is written to the first block of the hard disk by the **fdisk** utility. On a non-partitioned device (e.g. floppy) the OS loader will be the primary (and only) bootstrap loader. To write only the loader (and not wipe the filesystem clean) you must also specify the **-b** option.
-f *bootfile*	Preload **.boot** on the disk with the contents of the specified boot image.
-h	Initialize the hard disk. You can't initialize a hard disk unless you specify this option.
-l *label*	("el") *Deprecated*. Write volume *label* to disk after initializing.
-L *label*	*Deprecated*. Write only the volume *label* to disk.
-m *message*	Replace the message the OS displays when booting from disk with *message*.
-O	Use the old QNX bootstrap loader. The old loader loads at (real mode) 0x60:0, always. The newer loader looks for a signature byte in the beginning of the OS image to determine if it is old or new, and loads at 60h:00 or 80h:00 as appropriate. The start address for new images is 0h relative to the load

address, while the start address for old images is 20h relative to the load address.

-p Pause for a keystroke before continuing.

-q Be quiet; don't echo or question.

-r Write only the root block to disk.

drive Drive on which the hard disk or diskette is to be initialized (e.g. **/dev/fd0**, **/dev/hd0t77**, etc.).

Description:

The **dinit** utility initializes a formatted diskette or hard disk so that it may be used as a QNX filesystem. The default values are determined from the current configuration of the specified drive.

If the disk is a hard disk, you need to specify the **-h** (hard) option. This option helps protect you against typing errors that might cause **dinit** to initialize your hard disk. To initialize a hard disk, you must be the superuser.

Once you initialize a hard disk with **dinit**, you should use the **dcheck** utility to remove any bad blocks from the disk allocation bitmap. For example:

```
dinit -h /dev/hd0t77
dcheck -m /dev/hd0t77
```

To initialize a disk on another node, you must explicitly indicate the disk via a **//***node* prefix.

Disk labeling (**-l** and **-L** options) is a deprecated feature. These options will be eliminated in a future release.

When **dinit** initializes a disk, it writes a loader in the first block. If the disk is a floppy diskette, the loader is the bootstrap loader, else it is the secondary (or partition) loader. If you feel the need to rewrite the loader without re-initializing the disk, you may specify the **-b** option.

The **-m** option lets you change the message the OS displays when booting from disk. Normally, the message is:

```
Press ESC to boot alternate OS.
```

Your new message must include a trailing period. The OS will display the specified text up to the first period encountered. For example, if *message* is "**Booting ...**", the printed message will be:

```
Booting .
```

Your new message may contain up to 30 characters plus the trailing period. You can specify the minimum message of "**.**" by entering:

```
-m.
```

Examples:

Initialize a hard disk:

```
dinit -h /dev/hd0t77
```

Initialize a floppy disk:

```
dinit /dev/fd0
```

Pause before initializing hard disk:

```
dinit -hp /dev/hd0t77
```

Initialize floppy disk on node 3:

```
dinit //3/dev/fd0
```

Files:

All files are lost on the specified disk, unless **-b** is used.

Exit status:

0 Successful.

>0 An error occurred.

Caveats:

Don't use the **-r** option unless you know exactly what you're doing. You use the **-r** option only after a disaster has destroyed the first few blocks of your disk (e.g. a power failure occurred while the disk was being updated). In order for any damage to be repaired, you must follow **dinit -r** with this command:

chkfsys *drive*

See also:

chkfsys, **dcheck**, **fdformat**, **fdisk**, **mkdir**

dirname

© 2005, QNX Software Systems

Return directory portion of pathname (POSIX)

Syntax:

dirname *string*

Options:

string A string representing a valid pathname.

Description:

The **dirname** utility returns a portion of the *string* operand to standard output. The *string* operand represents a valid pathname whose format is:

directory_pathname/*base_filename*

The **dirname** utility writes the *directory_pathname* component to standard output.

If *string* is **//**, then **//** is returned. Any other string consisting entirely of slash characters causes a single slash to be returned.

You usually use the **dirname** utility inside command substitution marks, **$(**...**)**, within shell procedures.

Examples:

Command:	Output:
dirname .	**.**
dirname ..	**.**
dirname ../.	**..**
dirname /usr/src/prog.c	**/usr/src**
dirname /usr/src/	**/usr**
dirname .../[fred]	**...**

Files:

The **dirname** utility writes the directory name string derived from the command-line parameters to its standard output.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

basename

disktrap

© 2005, QNX Software Systems

Autodetect disk controller and start appropriate driver (QNX)

Syntax:

Print command line necessary to start driver:

disktrap [*options***] [query]**

Start driver:

disktrap [*options***] start**

Options:

-v Verbose output to help debugging failed detection.

-x *type* Exclude scan for this type of card. Cards are scanned for in the order the valid types are listed, below:

ps2, **ncr**, **aha2**, **aha7**, **amd**, **aha4**, **ide**

Description:

The **disktrap** utility scans the machine it is run on for disk controller hardware. Depending on the mode it is in (**query** or **start**), it will either report the correct driver to use for the hardware as a message to the standard output (**query**), or will start the appropriate driver (**start**).

The set of disk controllers **disktrap** can scan for may change as new controllers are supported by QNX drivers. Check the usage message for a definitive list (**use disktrap**). The set will include at least the following hardware types:

amd
ps2
ncr
aha2
aha7
aha4

ide

You can run **disktrap** only when logged in as **root**.

For driver:	Location:
Fsys.ps2scsi	checks addresses 0x96, 0x100 0x101, then 0x94, 0x102
Fsys.ncr8scsi	uses PCI BIOS
Fsys.aha2scsi	checks addresses 0x140, 0x340
Fsys.aha4scsi	checks addresses 0x130, 0x134, 0x230, 0x234, 0x330, 0x334
Fsys.aha7scsi	checks PCI BIOS, then addresses 0x1c00, 0x2c00, 0x3c00, 0x4c00, 0x5c00, 0x6c00, 0x7c00, 0x8c00, 0x9c00, 0xac00, 0xbc00, 0xcc00, 0xdc00, 0xec00, 0xfc00
Fsys.ide	checks addresses 0x1f0, 0x170
Fsys.ata	checks addresses 0x1f0, 0x170
Fsys.amdscsi	uses PCI BIOS

Files:

The standard input is unused.

Error messages and **-v** status messages are written to the standard error.

In **start** mode, the standard output is unused. In **query** mode, a message indicating the appropriate driver to run will be written to the standard output.

© 2005, QNX Software Systems

Exit status:

>0	Failed to detect card and/or start driver.
0	Successfully detected card/started driver.

See also:

Fsys, **Fsys.***, **inputtrap**, **mousetrap**, **nettrap**, **pcmciatrap**, **ramtrap**

Syntax:

ditto [-c *command_char***] [-f] [-k] [-n** *node***]**
[-q] [-t *console***] [***device_pathname***]**

Options:

-c *command_char*

Enter command mode on this character (hex value). Default is 05 (or Ctrl – E).

-f Follow "active" console.

-k Enable local keyboard.

-n *node* View the console on this node.

-q Quiet operation; don't notify target being watched.

-t *console* View this console number (0-10).

device_pathname

The pathname of a console device (e.g. **//1/dev/con1**).

Description:

The **ditto** utility allows a user to attach their local console or terminal to a remote console. While attached, the local screen reflects the contents of the remote screen, and any changes made to the remote screen are updated on the local one. When the keyboard is enabled, key strokes entered by the user of **ditto** are "injected" into the remote console, as if typed directly by the remote user.

The **ditto** utility can monitor only *consoles*, typically memory mapped devices. Any device driver which supports the console-specific functions (defined in the Watcom C *Library Reference for QNX*), can support **ditto**. Currently, this restricts the "ditto-able" devices to those provided by **Dev.ansi**, **Dev.con** and

wterm (the QNX windows terminal program). Terminal devices provided via ptys e.g. **xterm** (X-Windows System program) and **pterm** (Photon terminal program), are not ditto-able, though you may ditto a console *from* these devices.

You can use **ditto** from a **pterm** window in Photon to view a text-mode console; however, to view another **pterm** session you will need to use **phditto** instead of **ditto**. See **phditto** in the *Photon User's Guide*.

You must have appropriate permissions for the tty that you are attempting to use **ditto** on. If **ditto** fails on a tty, it may be because you do not have read permission for the target device.

While **ditto** is displaying the remote console on your screen, the cursor will be displayed as a blinking rectangle as an indicator that you are viewing a **ditto** session rather than one of your own local sessions. If you are using **ditto** from a tty which does not support a programmable cursor, the cursor will remain unchanged.

To exit from **ditto**, the command sequence Ctrl – E **q** can be given. (Press and hold the control key while typing an E. Release both keys, then type a **q**.)

Remote Typing

Upon entry, **ditto** will be in VIEW ONLY mode unless the **-k** command-line option was specified. This means that you are unable to interact with the remote console and anything you type at the keyboard will be discarded.

To enable the keyboard for remote typing you must enter the command character (Ctrl – E) followed by a **k** (see the Commands section).

Once remote typing is enabled, whatever you type is entered into the remote console. This facility allows two, or many, people to share any console on the network.

Console Sizes

In QNX the actual dimensions of a console are flexible depending on the type of video display hardware installed and console font or, in the case of **wterm**, the console dimensions specified by the user. For example an EGA can have 43 vertical lines while VGA can support up to 50 lines. (See **cfont** for information on using smaller console fonts, and **stty** for information on switching between fonts.)

As **ditto** attaches to different consoles, each with a potentially different number of displayed lines, your screen will be adjusted to match the number of lines on the remote console as closely as possible. (This means **ditto** automatically switches your screen to 43 lines if you attach to a console which has 43 lines ...).

If the entire contents of the remote console cannot be displayed locally (e.g. when you are not running **ditto** from a console), **ditto** will display only the portion of the remote console containing the cursor.

Privacy/Security

Now that **ditto** can provide network and dial-up users with complete, transparent access to every screen in the network, some users may find it necessary to implement some form of privacy or security. A command called **noditto** provides this functionality. This command is run in the background (it will place itself in the background if the user does not specify an **&** after the command) and excludes all but the users and groups named in its command-line parameters from dittoing the node it is run on. See **noditto** for more information.

Commands

As a general rule, commands issued to **ditto** are preceded by pressing the Ctrl – E sequence. This sequence is known as the "command character" and is used to cause **ditto** to accept whatever is typed next as a command to **ditto** instead of data to be entered into the attached remote console. If a Ctrl – E needs to be entered into the remote application, the Ctrl – E character should be typed twice.

You can also use the **-c** *command_char* option to change the command character to something other than Ctrl – E.

After pressing the command character, **ditto** will display a prompt at the top left corner of the screen. Pressing a "?" at this point will cause **ditto** to display a command menu containing a list of available commands as well as the current status. If the user is unsure of what command to issue to the first prompt, **ditto** will also automatically display the command menu after a 3 second pause. The menu displayed by **ditto** is of the form:

```
--------------Commands--------------

 0    View initial console
 1-9  View specific console
 a    View active console
 A    Make current console active
 f    Follow (track) active console
 k    Turn keyboard on/off
 n    Display next console
 p    Display previous console
 q    Quit
 r    Reboot target node
 u    Update (repaint) screen
 <    Reduce number of lines
 >    Expand number of lines
 ?    Display help file
------------------------------------
```

Pressing the "?" key will start the builtin **ditto** help system.

Ditto commands

Here's an alphabetical list of commands:

0 View initial console.

This command causes **ditto** to switch to the console that it was attached to when it was initially invoked.

1-9 View specific console.

Pressing the number 1 through 9, will cause **ditto** to lock onto the specified console of the remote node. In much the

same manner, as the remote user will be typing Ctrl – Alt – 1 through Ctrl – Alt – 9 to switch between various consoles, a person using **ditto** can press the command character followed by a 1 through 9 to view a specific console. Once **ditto** has been *locked* to a particular console in this manner, it will always keep viewing it, even if it becomes inactive (i.e. not displayed on the remote node's monitor).

This mode is the default if **ditto** was invoked with the form:

ditto //*node***/dev/con***X*

See the "View active console" command if you wish instead to always view the active (displayed) console on a remote node.

a View active console.

This command will inform **ditto** that you wish to view the active (displayed) console of a remote node. If the user on the remote console switches virtual consoles, **ditto** will *not* follow the switch to the newly active console.

See the "Attach Specific Console" command if you wish instead to always view a specific console of a node.

A Make current console active.

Cause the console currently being viewed in **ditto** to be made the currently-displayed console on the machine being ditto'd.

f Follow (track) active console.

This command will cause **ditto** to follow the active console on the machine being ditto'd — i.e. if the user at that console switches virtual consoles, the remote **ditto** display will also switch and display the new active console.

This mode is the default if **ditto** was invoked with the form:

ditto *user_name*

© 2005, QNX Software Systems

k Keyboard on/off.

This command turns the keyboard on or off to enable or disable remote typing. By default, typing is off, causing **ditto** to discard any characters typed into the keyboard.

Having typing off can be useful for monitoring remote consoles that you don't want to accidentally type into. When typing is turned on, any data typed at the keyboard is also typed into the remote console. Note that both the remote keyboard and your keyboard can type into the same remote console.

Since typing the command character (Ctrl – E) causes **ditto** to enter command mode, if you want it to be entered into the remote console you need to type the command character twice.

Note that QNX compose key sequences, function keys and the BREAK key are completely operational.

n Display Next Console.

This command is used to cause **ditto** to transfer the attachment to the next sequential console on the remote node. Thus, this command can be used to quickly step through all the available remote consoles. Afterwards, the "View active console" command can be used to resume display of the currently displayed console again.

p Display Previous Console.

This command is used to cause **ditto** to transfer the attachment to the previous sequential console on the remote node. Thus, this command can be used to step backwards through all the available remote consoles.

q Quit.

This command will cause **ditto** to exit.

r Reboot.

This command will cause **ditto** to reboot the node being viewed. This command will operate only if you are running **ditto** as **root**.

u Update (repaint) screen.

This command is used to cause **ditto** to completely rewrite the entire screen. This can be useful if a noisy modem line has caused the local copy of the remote console to become incorrect.

< Reduce number of lines.

This command reduces the number of lines on the remote screen, as if the user there had entered Ctrl – Alt – **<**.

> Expand number of lines.

This command expands the number of lines on the remote screen, as if the user there had entered Ctrl – Alt – **>**.

? Display help file.

This command will cause **ditto** to display its online help file.

Examples:

View the first console on node 1:

```
ditto -n1
```

View console 3 on node 1 (**//1/dev/con3**), and enable keyboard control:

```
ditto -k -n1 -t3
```

View console 2 on node 1 by name.

```
ditto //1/dev/con2
```

© 2005, QNX Software Systems

View window 3 on a target running **Dev.win**: (**Dev.win** is the console driver for QNX Windows' **wterm**)

```
ditto //1/dev/win3
```

Files:

- **/usr/lib/ditto.hlp**
- Contains the ditto online help.

Exit status:

0 **ditto** was run successfully.

>0 An error occurred.

Caveats:

When a session is terminated on a console **ditto** is monitoring, **tinit** will not be able to bring up a new **login** until **ditto** detaches from that console.

See also:

Dev.ansi, **Dev.con**, **noditto**

Dosfsys

DOS filesystem manager

Dosfsys has been deprecated. Use the Fatfsys utility instead, which offers full support for FAT32, long filenames, and more.

Syntax:

Dosfsys [-S|-s] [-eLmt] [-T *timeout***]**
[*dos_drive***=***qnx_drive***[,R]]... &**

Dosfsys -i [-n *node***] [***dos_drive_path***]...**

Dosfsys -o [-n *node***]**

Dosfsys -x [-n *node***]**

Options:

-e	Set the execute bits for user, group, and other for all DOS files.
-i	Get drive information from the **Dosfsys** manager on the specified node. If no node number is specified, the local **Dosfsys** manager is queried.
-L	Allow Windows 95 long filenames to be read.
-m	Suspend adoption of the indicated drives; display only the drives that would be adopted.
-n *node*	The node the **Dosfsys** manager resides on.
-o	For each device, display the names of any files that are currently open.
-s	Scan the **/dev** directory for drives to adopt.
-S	Same as **-s**, except the drives are adopted as read-only.
-t	Translate and truncate QNX filenames.

-T *timeout*	Timeout for internal cache invalidation for removable media. (In seconds; default and minimum is 2 seconds).
-x	Cause the **Dosfsys** manager on the specified node to exit. If no node is specified, the local **Dosfsys** manager is used.
dos_drive	A DOS drive letter, ranging from **a** to **z**.
dos_drive_path	The full name of the adopted DOS drive (defaults: **/dos/a** for **a** drive, **/dos/b** for **b** drive, etc.)
qnx_drive[**,R**]	The name of the QNX drive or file to open (e.g. **/dev/hd0t4**). The drive or file can be local or it can reside on another node in the network. If the optional **,R** is specified, the DOS drive will be mounted as a read-only drive.

Description:

The **Dosfsys** filesystem manager provides transparent access to DOS files and directories that reside on a DOS disk. **Dosfsys** can support up to eight drives.

The **Dosfsys** manager uses block-special files hosted by the **Fsys** or **Blkfsys** managers for its access to real devices. One of these managers and the appropriate driver(s) must be running before **Dosfsys** is started.

You can create files on DOS disks, read them, write to them, and delete them with QNX programs. You can also create, read, and delete directories on DOS disks with standard QNX utilities, such as **mkdir**, **ls**, and **rmdir**. Standard QNX utilities will work with DOS files, provided that the DOS file structure allows for the functionality required by the utility.

Your own programs will also be able to process DOS files just as they process QNX files, by using the standard I/O functions such as *open()*, *read()*, *write()*, *close()*, *seek()*, etc. When you read DOS directories, they'll be presented to you in QNX format.

The **-i** option lets you get information about the currently adopted DOS drives. The **-o** option lets you see the names of any files that are currently open. The **-x** option terminates the **Dosfsys** server — no new *open()* requests will be accepted and the server will terminate once all active files (i.e. files that are still open) are closed.

If you don't specify **-i**, **-o**, or **-x**, **Dosfsys** will start up and try to adopt the specified drives.

To start or terminate the **Dosfsys** server, you must be logged in as the superuser (**root**).

Unless otherwise specified on the command line, **Dosfsys** will adopt up to eight drives. DOS primary partitions (**/dev/hd0t1**, **/dev/hd0t4**, **/dev/hd0t6**, ...) and extended partitions (**/dev/hd0t1.1**, **/dev/hd0t1.2**, ...) are mounted starting at drive **C** (**/dos/c**). Floppy drives are mounted as follows:

Raw floppy device	Dosfsys mount point
/dev/fd0	**/dos/a**
/dev/fd1	**/dos/b**

The Microsoft Windows 95 release has added support for long filenames that are not limited to the traditional DOS "*cccccccc*.*xxx*" filename convention. **Dosfsys** has support to handle existing Windows 95 files with "long" filenames when the **-L** option is specified. **Dosfsys** does *not* support *creating* files with long filenames — the file must already exist.

Examples:

Scan the **/dev** directory and mount the corresponding drives:

```
Dosfsys &
```

or

```
Dosfsys -s &
```

The above forms are the norm for most environments.

Adopt **drivea** as **/dos/a** (read-only) and **//8/disks/drivec** as **/dos/c**:

```
Dosfsys a=drivea,R c=//8/disks/drivec &
```

Adopt **/dev/fd0** as **/dos/a** and **/dev/hd0t4** as **/dos/c**:

```
Dosfsys a=/dev/fd0 c=/dev/hd0t4  &
```

Files:

Dosfsys closes its standard input, standard output, and standard error after it completes its startup initializations.

Dosfsys adopts files in the pathname prefix tree of the node it is being run on. Usually these will appear under the directory **/dos**. See description for more detail.

Exit status:

If an error occurs at startup, **Dosfsys** displays an appropriate error message, then exits with one of the following exit codes:

1 Exit due to signal.

2 Can’t register name.

3 Can’t create proxy.

4 Can’t register drive prefix.

5 Can’t set up timer.

6 Invalid command-line option(s).

7 Error mounting drive.

8 Not enough memory for buffers.

9 Invalid drive specifier.

10 No drive specified on the command line.

11 Can't attach to file descriptor table.

Caveats:

Dosfsys will not work with media that has a sector size larger than 512 bytes.

See also:

Blkfsys, **Fatfsys**, **fdisk**, **Fsys**, **Fsys.***, **textto**, **tr**

QNX *Installation & Configuration*

du

© 2005, QNX Software Systems

Estimate disk space usage (POSIX)

Syntax:

du [-a|-s] [-ikpqx] [*file***...]**

Options:

-a Generate a report for each file in the directory tree. If you don't specify this option, **du** makes a report for each directory only. The report shows the total space allocated to all files under the directory, including the directory itself.

-i Count extra blocks used for the inode indexing of multi-extent files. (Slower!)

-k Report the space figures in Kbytes (the default is 512-byte blocks).

-p Report the space figures in bytes (the default is 512-byte blocks). Also, ensure that **du** generates error messages when it can't process existing files (unless the **-q** option is specified).

-q Be quiet; suppress error messages when **du** cannot provide statistics on files, or cannot read directories.

-s Give the total figures for each of the specified files, rather than the totals for any subdirectories.

-x Don't span device boundaries (used to determine how much space on a particular device is consumed by a directory tree).

file The pathname of a file whose size is to be displayed. If you don't specify any files, the current directory is used; **du** behaves as if the filename dot (**.**) were given.

Description:

The **du** utility prints the size of the file space allocated to the specified files. If you name a directory, all files in that directory are reported; subdirectories are traversed recursively. If a file has multiple links, the space allocated to that file is counted only once.

The **du** utility doesn't take into account the filesystem space taken up by persistent pregrown files, i.e. files which have had space in excess of the file size pre-allocated in anticipation of future growth. Pregrowing is often done by the filesystem for files which have been appended to. Filesystems prior to 4.24 didn't support this feature. When using current filesystems which do support persistent pregrown files, you can disable this feature by mounting the disk with the **mount** utility's **-g** flag.

The space figures are displayed in 512-byte blocks by default. If you want **du** to print the size in bytes, you specify the **-p** option. The sizes output by **du** when you specify the **-p** option are accurate with one exception. The numbers may be slightly higher than expected because they include extent blocks that are part of filesystem overhead associated with the file, but don't contain actual data.

If you specify nondirectories, they aren't listed unless you specify the **-a** option.

Files with multiple links are reported only once.

Examples:

Estimate disk space consumed by the contents of **/tmp**, in kbytes:

```
du -k /tmp
```

Estimate the total space occupied by the contents of the current directory:

```
du -s
```

Files:

All results are written to the standard output. Errors may result in diagnostic messages to the standard error. Standard input isn't used.

© 2005, QNX Software Systems

The **du** utility will obtain directory information pertaining to the file(s) and directories named on the command line and, for each file that is a directory, will recurse through the filesystem obtaining directory information about the files and directories contained therein.

Exit status:

0 Successful completion.

>0 An error occurred. This doesn't include failure to read files or directories.

See also:

df

Syntax:

dumper [-d *path***] [-p** *pid***] [-s] &**

Options:

-d *path*	The name of the directory in which postmortem dump files are written. If not specified, the default is the user's home directory.
-p *pid*	Save a dump file for this process if it terminates for any reason. Don't save a dump file for any other process.
-s	Create sparse dump files (produces much smaller dump files).

Description:

The **dumper** utility runs in the background and provides a postmortem dump service for all processes. Whenever a program terminates abnormally, a dump of the current state of the program is written to disk. The dump filename is the same as the program name with a **.dmp** extension. For example, if the program name is **experiment** the dump will be written to the **/home/***userid***/experiment.dmp** file.

You can use the **-d** option to force all dumps into a single directory rather than into the invoking user's home directory.

The **-p** option lets you monitor a particular process. You can run multiple copies of **dumper**, each monitoring a different process.

A dump file may be examined using a debugger:

wd -tr=pmd *programname***.dmp**

The dump file format is described in the **/usr/include/sys/dumper.h** C header file.

A program may terminate in one of two ways. It may exit cleanly under its own control, returning an exit status, or it may be forcibly

terminated by the receipt of a signal it is not prepared to handle. In the latter case, **dumper** will write a dump file for the following set of signals:

Signal	Description
SIGABRT	program-called abort function
SIGBUS	parity error
SIGFPE	floating-point error or divide by zero
SIGILL	illegal instruction executed
SIGSEGV	segmentation violation

You can force the dump of a running program by setting one of the preceding signals, assuming that the program is not masking or handling the signal itself. For example:

Force a dump using the **kill** command and a process ID (pid):

```
kill -sigabrt pid
```

Force a dump using the **slay** utility and the process name:

```
slay -s sigabrt process_name
```

Examples:

Start dumper, with dump files to be written to the home directory of the user who was running the process being dumped:

```
dumper &
```

Start dumper, dump files to be placed in the directory **/home/dumps**:

```
dumper -d /home/dumps &
```

Files:

After initialization has been completed, **dumper** closes its standard input, standard output and standard error file descriptors. If an error is encountered during initialization, a diagnostic message is written to standard error before **dumper** terminates. Other than this, no output is produced to these files.

The **dumper** utility will create dump files in the home directory of the user who was running the program being dumped. These files will be named **/home/***userid***/***program***.dmp**. The directory may be different if the **-d** *path* option was specified on the command line.

Exit status:

The **dumper** utility normally does not terminate. However, it may terminate if it encounters an error on startup (for instance, if it was not run by **root**) or if it receives a signal.

0 A signal was received and dumper shut down successfully.

1 An error was encountered on startup (not run by **root** or bad command-line options).

See also:

kill, **slay**

WATCOM Debugger User's Guide

echo

© 2005, QNX Software Systems

Write arguments to standard output (POSIX)

Syntax:

echo [*string***...]**

Options:

string A string to be written to standard output.

Description:

The **echo** command is present both as a shell built-in and as a stand-alone executable which can operate without the availability of the system shell. Both versions behave in a similar manner.

The **echo** utility writes its arguments, followed by a **newline** character, to standard output. If there are no arguments, only the **newline** character is written.

The **echo** utility currently interprets backslash escape sequences within the string in a fashion similar to the SysV **echo** command. However, POSIX dictates that the **echo** utility must not interpret or modify the string in this manner. A future version of **echo** will not support the interpretation of backslash escape sequences and the use of this feature should be avoided.

The following backslash escape sequences are currently supported within *string*:

Sequence:	Meaning:
\a	Write an **alert** character (the bell).
\b	Write a **backspace** character.
\c	Suppress the **newline** character that otherwise follows the final argument in the output. All characters following the **\c** in the arguments are ignored.

continued...

Sequence:	Meaning:
\f	Write a **form-feed** character.
\n	Write a **newline** character.
\r	Write a **carriage-return** character.
\t	Write a **tab** character.
\v	Write a **vertical tab** character.
****	Write a **backslash** character.
\0*num*	Write an 8-bit value that is the 1-, 2-, or 3-digit octal number *num*.

Examples:

Echo the string “Hello, Mother\nHello, Father” to the standard output:

```
echo 'Hello, Mother\nHello, Father'
```

Note that **echo** will append a final trailing **newline** to its output, e.g.:

```
$ echo 'Hello, Mother\nHello, Father'
Hello, Mother
Hello, Father
$
```

Files:

The **echo** command writes a modified version of its command-line parameters to standard output. Standard input and standard error are not used.

© 2005, QNX Software Systems

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

sh

Erase Flash partitions or format an embedded filesystem (QNX)

Syntax:

efsinit [-d] [-l] [-s *num***]** *path*

Options:

-d	Format as a DOS-compatible filesystem, that is, with 8.3 filenames.
-l	Format as a POSIX-compatible filesystem.
-s *num*	The number of spare blocks (the default is 1).

Description:

The **efsinit** utility has two functions:

- to erase the entire Flash device
- to construct an empty embedded filesystem

The utility decides which function to perform based on the path you specify. The path must have been created by one of the Efsys drivers.

- If you specify a raw device, the utility simply erases blocks, but doesn't write any filesystem information.
- If you specify an embedded filesystem mountpoint, the utility erases all blocks within the partition, and then writes an empty filesystem to the partition.
- If you try to erase memory devices that have no erase capability (e.g. ROM), nothing happens.

© 2005, QNX Software Systems

If you've already formatted an embedded filesystem partition on the Flash device, be careful about erasing the entire Flash, since you'll lose the block-usage information stored in the partition. You may need this information if you're using Flash devices that have a low maximum erase-cycle count.

Currently all read/write filesystems require a spare block independent of the memory devices used to implement the Flash.

Examples:

Erase all data on a raw Flash device:

```
efsinit /dev/skt1
```

Erase and format an empty embedded filesystem with one spare block:

```
efsinit /efs1p1
```

Efsys.365sl

Embedded filesystem driver for Intel 82365 PCMCIA adapter (QNX)

Syntax:

Efsys.365sl [-a *addr***] [-d** *name***] [-i]**
[-m *mountpoint***] ...**

Efsys.365sl [-a *addr***]**

Options:

-a *addr* The I/O port address of the adapter. (Default: **0x3E0**).

-d *name* A raw device name (the default is **skt**).

-i Ignore CIS information

-m *mountpoint*

The mountpoint for the embedded filesystem (EFS) partitions.

Description:

Efsys.365sl provides embedded filesystem support for PCMCIA memory cards that are connected to the host system through a PCMCIA adapter. (The supported adapter is based on the Intel 82365SL PCMCIA controller device and can handle up to two sockets.)

The **Efsys.365sl** accesses the PCMCIA card using the low-level socket services interface and won't interoperate with other PCMCIA drivers.

The driver uses the following default filenames:

/dev/skt1 raw device special file for EFS socket 1

/dev/skt2 raw device special file for EFS socket 2

© 2005, QNX Software Systems

/dev/skt*X***img***Y*

raw device special file for each image partition on each EFS, where *X* is the logical socket, and *Y* is a unique partition number

/efs*X***p***Y* embedded filesystem mountpoint for each EFS partition on each EFS, where *X* is the logical socket, and *Y* is a unique partition number

To change the raw device names, use the **-d** option. To change the mountpoints of the EFS partitions, use the **-m** option. You can specify multiple **-m** options.

Efsys.file

Embedded filesystem in a file

Syntax:

Efsys.file [-d *name*]**[-f** *basename*]
[-m *mountpoint*] **... [-s** *num*]

Options:

-d *name*	A raw device name (the default is **skt**).
-f *basename*	The basename of the file to store data in (the default is **socket**).
-m *mountpoint*	The mountpoint for the embedded filesystem (EFS) partitions.
-s *num*	The number of sockets (the default is 1).

Description:

Efsys.file operates similarly to the other Efsys drivers but stores the filesystem within a file. This lets you create a filesystem using normal tools (e.g. **efsinit**, **mkcis**, **mkdir**, **cp**, etc.) and then program the resulting image into ROMs for use in a target system.

Don't use the **mkcis -2** option if you're using **Efsys.file**. **Efsys.file** can't read the level 2 structures.

The driver uses the following default filenames:

/dev/skt*X* raw device special file for each EFS socket, where *X* is the logical socket.

/dev/skt*X***img***Y*

raw device special file for each image partition on the EFS, where *X* is the logical socket, and *Y* is a unique partition number

/efs*X***p***Y* embedded filesystem mountpoint for each EFS partition on the EFS, where *X* is the logical socket, and *Y* is a unique partition number

© 2005, QNX Software Systems

To change the raw device names, use the **-d** option. To change the mountpoints of the EFS partitions, use the **-m** option. You can specify multiple **-m** options.

Efsys.file stores the data for each logical socket within a file named **socket.***X*, where *X* is the number of the logical socket. You can change this name with the **-f** option.

Before using the contents of the data files, you should slay **Efsys.file**.

Examples:

Start **Efsys.file**, with output to **socket.1**:

```
Efsys.file
```

Start **Efsys.file**, with two sockets called **rom.1** and **rom.2**:

```
Efsys.file -s 2 -f rom
```

See also:

romqnx

Efsys.pcmcia

Generic embedded filesystem driver options

Syntax:

Efsys.pcmcia [-d *device_number***]**
[-i] [-v]
[-m *mountpoint***] ...**

Options:

-d *device_number*

Create raw devices starting from **/dev/skt***device_number*. For example, if you specify **-d 10** or **-d10**, the first PCMCIA socket is called **/dev/skt10** and its mountpoint is **/efs10p***Y*.

-i Ignore CIS information

-m *mountpoint*

The mountpoint for the embedded filesystem (EFS) partitions

-v Be verbose; display a message if an error occurs.

Description:

The **Efsys.pcmcia** driver supports hot swapping. All access to PCMCIA cards is handled by **Pcmcia.generic** (see the QNX *Utilities* reference), so **Efsys.pcmcia** can interoperate with other PCMCIA drivers.

Efsys.pcmcia can be started only if the **Pcmcia.generic** server is already running.

Efsys.pcmcia uses the following default filenames:

/dev/skt1 raw device special file for EFS socket 1

/dev/skt2 raw device special file for EFS socket 2

© 2005, QNX Software Systems

/dev/skt*X***img***Y*

raw device special file for each image partition on each EFS, where *X* is the logical socket, and *Y* is a unique partition number

/efs*X***p***Y* embedded filesystem mountpoint for each EFS partition on each EFS, where *X* is the logical socket, and *Y* is a unique partition number

To change the raw device names, use the **-d** option. To change the mountpoints of the EFS partitions, use the **-m** option. You can specify multiple **-m** options.

Syntax:

Efsys.ram **[-b** *blockshift***] [-f]**
[-m *addr*,*size*,*mount***[**,*raw_device***]]...**
[-r]

Options:

-b *blockshift* Specify the block size used by the filesystem. The actual block size used will be 2^*blockshift*. Values used should fall within the range 12 through 23. The default is 15.

-f Strictly emulate flash (testing only).

-m *addr*,*size*,*mount*[,*raw_device*]

The **-m** option should always be specified. Its arguments describe the address and size of the physical memory area to be utilized, and the mountpoint through which the filesystem is to be accessed.

If an address of 0 is specified, **Efsys.ram** will allocate an area in available system RAM. If an address is specified, the memory must be outside the system RAM known to the operating system. Normally this would be a bank of SRAM or EPROM. The size must be non-zero and may optionally be followed by a "**k**", denoting kilobytes, or an "**m**", denoting megabytes. The mountpoint is the pathname mount point the filesystem should appear at.

The optional *raw_device* argument specifies a mount point by which the raw device data may be accessed (normally it is a pathname under **/dev**). If no *raw_device* argument is specified, there will not be a pathname through which the raw RAM data may be accessed.

Multiple **-m** options may be specified.

-r The filesystem is to be read-only.

Description:

Efsys.ram provides an embedded filesystem using RAM or ROM. It can be used with battery backed up SRAM to provide a read/write filesystem or with EPROMS to provide a read-only filesystem. The filesystem has the same limitations as the other embedded filesystem drivers. Since this driver is only used to access a filesystem, there is no CIS information stored at the beginning of the image.

The raw mount point which may be optionally specified as part of the **-m** option is useful for capturing an image of the filesystem which can be then be burned into ROM.

Note that like a flash filesystem, the RAM filesystem must use a spare block. In **Efsys.ram** this is basically wasted space and should be minimized. This can be done by reducing the blocksize that the embedded filesystem uses. There is however a tradeoff to be made. The smaller the block size is, the greater is the overhead required to manage the blocks. Normally a 4K or 8K block is the smallest usable blocksize.

If **Efsys.ram** is pointed at an existing filesystem image, or an image is loaded into the raw mountpoint, the blocking size used to create that image must match the blocking size specified to **Efsys.ram**.

Care should be used when specifying a non-zero address, since if this points at system RAM that is in the OS memory pool the driver may interfere with the operation of other processes or the operating system itself. The **Efsys.ram** filesystem does not check that the address specified does not conflict with the OS.

Examples:

Start a 1M ramdisk from the OS memory pool with a mount point of **/ramdisk** and a raw device **/dev/ramimage**:

```
Efsys.ram -m 0,1024K,/ramdisk,/dev/ramimage &
```

Start a ROM filesystem using ROMs occupying 128k of space starting at D000:0, and used a reduced block size of 4K:

```
Efsys.ram -b12 -m 0xd0000,128K,/rom
```

egrep

© 2005, QNX Software Systems

Extended regular expression grep (UNIX)

Syntax:

egrep [-chilnqsvx] [-e *expression***|-f** *expression_file***]...**
[*file***...]**

egrep [-chilnqsvx] *expression* **[***file***...]**

Options:

See **grep**.

Description:

The **egrep** utility does an extended regular expression **grep** (equivalent to **grep -E**). See **grep** for details.

Exit status:

0 Lines were found matching the expression provided.

>0 An error occurred or no matching lines were found.

See also:

grep, **diff**, **find**, **sort**

Syntax:

elvis [*options***]... [+***command***]** *file***...**

Options:

-c *command*	Begin editing by executing this **ex** command.
-e	Start up in colon (**ex**) command mode.
-i	Start up in input mode.
-R	Set read-only mode to prevent accidental overwriting of files.
-t *tag*	Start editing at the given tag. (See the **ctags** utility.)
-v	Start up in visual command mode.
+*command*	Begin editing by executing this **ex** command.
file	A pathname of a file to be edited.

Description:

The **elvis** utility is an interactive fullscreen editor that is compatible with the Unix/POSIX **vi** editor. The **vi** utility in QNX is a link to **elvis**.

In **elvis**, changes are buffered and are written to the file only upon request. Since a temporary file is used for storage, **elvis** can edit files larger than the amount of memory available on the machine it is run on.

There are two major command modes:

- **vi**sual mode
- **ex** mode (also known as colon mode)

© 2005, QNX Software Systems

You can switch between modes. You'll probably use the visual command mode most of the time. This is the mode **elvis** normally starts up in.

In visual mode, the entire screen is filled with lines of text from your file (except the last screen line, which is reserved for status). You can view text and move around the file. Each keystroke is interpreted as part of a visual command. If you start typing text, it won't be inserted; instead, it'll be treated as part of a command. To insert text, you must first give an "insert text" command (see "Inserting text").

The **ex** mode is quite different. In this mode, **elvis** displays a "**:**" character on the bottom line of the screen as a prompt. You are then expected to type in a command and press Enter. The set of commands recognized in **ex** mode differs from those in visual mode, and are known as **ex** commands. A summary of these commands is found at the end of this man page. (A line-oriented editor, **ex** is a predecessor of **vi**, and the two share common functionality.)

The capabilities of **elvis** are described throughout the following sections:

1 Visual Mode

- Input Mode
- Operators
- Special Cases
- Named Buffers

2 Movement Commands

- Text Objects
- Cursor Movement
- Marking
- Tags

3 Inserting text

- Input Mode

4 Deleting, Yanking, Putting

- Copying
- Deleting
- Moving

5 Filters

6 Shifting text

7 Miscellaneous commands

8 Searching

- Patterns
- Search Commands
- Search Examples

9 Global & Substitute Commands

- Substitute Commands
- Ranges
- Global Commands
- Global Examples

10 Undo and Retrieving

- Undo
- Retrieving

11 Screen Commands

- Scrolling
- Screen Adjustments
- Status

12 File Manipulation

- Writing to a File
- Editing Other Files

© 2005, QNX Software Systems

- Reading In a File
- Leaving elvis

13 Escaping to a Shell

14 Abbreviations and Macros

- Macros
- Abbreviations

15 Options

16 **ex** commands

1. Visual mode

Most visual mode commands are one keystroke long. The following sections list the operation performed by each keystroke, and any necessary options or operands.

Most commands may be preceded by a decimal number. Usually, this number specifies how many times the command is to be repeated. Without this number, the command will in most cases execute just once.

Input mode

You cannot enter text into your file directly from visual command mode. Instead, you must first give a command that puts you into input mode. The commands to do this are:

A, **C**, **I**, **O**, **R**, **S**, **a**, **c**, **i**, **o**, **s**

Visual command mode looks a lot like text input mode. If you forget which mode you're in, simply hit Esc. If **elvis** beeps, you're in visual command mode. If **elvis** doesn't beep, you *were* in input mode — by hitting Esc you'll have switched to visual command mode. One way or another, after you hit Esc, **elvis** will be ready for a command.

Note that if the **showmode** option is set (this option described in the "Options" section below), the mode will be displayed in the lower right-hand corner as either **Command** or **Input**. Also note that each mode uses a different cursor shape, if supported by the output device.

Operators

Among its various commands, **elvis** has seven basic but powerful operators that let you change, delete, cut, paste, shift, or filter text regions. Although they're described below in the appropriate sections, you should note that, for the most part, they all share a common form:

*op***<***object***>**

Where *op* can be:

c Change

d Delete

y Yank

> Shift right

< Shift left

! Filter

One operator does not take an *object*:

p Put

© 2005, QNX Software Systems

The *object* operand specifies the range of text that the *op* operator will act upon. The *object* can be any movement command or pattern search. (See Movement Commands). In this way, the power of the operators can be seen. The **elvis** utility can act on words, sentences, regions up to a specified pattern-whatever range the *object* operand specifies.

When you give a movement command as an operand, you can specify an optional count [*n*] to multiply the effect of the operand (e.g. **d2w** to delete next two words). When a count isn't provided, it typically defaults to 1. See the section on Movement Commands.

Special cases

Operators are frequently used with lines. So, for simplicity, a few special forms of these commands operate only with line *object*s. The following list shows the syntax of these special forms. Note the count is optional, and precedes the operator (this differs from the above form):

[*n*]**cc** Change *n* lines

[*n*]**dd** Delete *n* lines

[*n*]**Y**
[*n*]**yy** Yank *n* lines

[*n*]**<<** Shift *n* lines to the left

[*n*]**>>** Shift *n* lines to the right

Named buffers

When you yank, delete, change, shift, or filter text, **elvis** saves the affected region in a single unnamed cut buffer (which you can recall by the **put** command). However, **elvis** also has 26 named buffers, *a* to *z*, which you can use to save blocks of text during an edit session. These buffers are often used when editing multiple files to move text around.

The previously defined operator forms can now be prefixed by **"a** through **"z** (represented below by **"<**a-z**>**) to indicate which cut buffer is to be used to store the modified text region:

"<a-z**>***op***<***object***>**
normal cases

"<a-z**>**[*n*]*op* special cases

When you use the uppercase letters to denote the named buffers, the *object*s are appended to the buffer. Thus adding the following forms:

"<A-Z**>***op***<***object***>**
normal cases

"<A-Z**>**[*n*]*op* special cases

2. Movement commands

The following movement commands provide a convenient means to position the cursor throughout the file being edited. Most can be preceded by an optional count to repeat the action. More importantly, these movement commands can be used as operands to the change, yank, delete, put commands to specify the range of action to be performed (see “Inserting text” and “Deleting, yanking, putting” sections).

Text objects

The following text objects are used in conjunction with the movement commands:

characters — A single character.

Sentences — A sentence is a sequence of text ending in one of the following characters, followed by two **space**s or a **newline**:

© 2005, QNX Software Systems

. ! ?

Paragraphs	A paragraph starts after an empty line or if any of the pairs of characters defined with the **:set pa=** option are found.
Sections	A section starts where any of the pairs of characters defined with the **:set se=** option are found.

Cursor movement

To move the cursor, you can use the keypad arrow keys; you can also use the H, J, K, and L keys.

[*n*]**j** [*n*]down-arrow [*n*]Ctrl – J [*n*]Ctrl – N	Move down *n* lines (next).
[*n*]**k** [*n*]up-arrow [*n*]Ctrl – P	Move up *n* lines (previous).
[*n*]**l** [*n*]right-arrow [*n*]spacebar	Move right *n* characters.
[*n*]**h** [*n*]left-arrow [*n*]Backspace	Move left *n* characters.
[*n*]**-**	Move to first non-blank character on *n*th line, in backward direction.
[*n*]**+**	Move to first non-blank character on *n*th line, in forward direction.
[*n*]**$**	Move to the end of the line.

^	Move to the beginning of the first word on the line.
0	Move to the left margin (first non-blank character) of current line.
[*n*]**|**	Move to the column specified by *n*.
[*n*]**w**	Move to the beginning of the next (or *n*th next) word.
[*n*]**W**	Move to the beginning of the next (or *n*th next) word that follows white space.
[*n*]**b**	Move to the previous (or *n*th previous) word.
[*n*]**B**	Move to the previous (or *n*th previous) word that's delimited by white space.
[*n*]**e**	Move to the end of the current or *n*th word.
[*n*]**E**	Move to end of the current or *n*th word that's delimited by white space.
[*n*]**G**	Move to the specified line (default is last line of file for a simple **G** command).
[*n*]**f<***char***>**	Move to the *n*th occurrence of *char* (in forward direction) on the current line. The cursor is placed at the matched character.
[*n*]**F<***char***>**	Move to the *n*th occurrence of char (in backward direction) on the current line. The cursor is placed at the matched character.
[*n*]**t<***char***>**	Move to the *n*th occurrence of *char* (in forward direction) on the current line. The cursor is placed just before the matched character.
[*n*]**T<***char***>**	Move to the *n*th occurrence of char (in backward direction) on the current line. The cursor is placed just before the matched character.

© 2005, QNX Software Systems

[*n*]**;**	Repeat previous **f**, **F**, **t**, or **T** command, in the same direction.
[*n*]**,**	Repeat previous **f**, **F**, **t**, or **T** command, in the opposite search direction.
%	Move to matching parenthesis, bracket, or brace; i.e.: **( ) [ ] { }**
[*n*]**)**	Move to the beginning of the next (or *n*th next) sentence.
[*n*]**(**	Move to the beginning of the current (or *n*th previous) sentence.
[*n*]**}**	Move to the beginning of the next (or *n*th next) paragraph.
[*n*]**{**	Move to the beginning of the current (or *n*th previous) paragraph.
[*n*]**]]**	Move to the beginning of the next (or *n*th next) section.
[*n*]**[[**	Move to the beginning of the current (or *n*th previous) section.
[*n*]**H**	Move to the top left position of screen. If *n* is specified, move the cursor to the beginning of the line *n* lines from the top of the screen.
[*n*]**L**	Move to the beginning of the last line on screen. If *n* is specified move the cursor to the beginning of the line *n* lines from the bottom of the screen.
M	Move to the beginning of the middle line on the screen.

Marking

m<*a-z***>**	Mark the current position with the character **<***a-z***>**.
'<*a-z***>**	Move to the beginning of the marked line.
`<*a-z***>**	Move to the exact position marked with the character **<***a-z***>**.
''	Move back to the beginning of the line where it was before the last "non-relative" move.
``	Move back to the exact position it was before the last non-relative move.

Tags

Tags can't be included as **<***object***>** operands for commands such as change, yank, put, delete, shift, and filter. Tags are defined in a file created by the **ctags** utility.

:ta *tag*	Edit the file containing *tag*. Position at *tag*.
Ctrl –]	The word at the cursor position is taken as the tag, and the editor finds the word as with the **:ta** command.

3. Inserting text

The following commands enter input mode, where the text you enter is put into the file, until you press Esc.

a	Append text after the current cursor position.
A	Append text at the end of current line.
i	Insert text before the current cursor position.
I	Insert text at beginning of current line.

© 2005, QNX Software Systems

o	Open a new line below line cursor is on, insert text there.
O	Open a new line above line cursor is on, insert text there.
c<*object***>**	Change the text between the current position and the position specified by the **<***object***>** (movement operand or pattern). An optional count can be specified to multiply the effect of the **<***object***>** operand (i.e. **c2w** to change next two words). If the range is within the current line, a **$** will be displayed at the end of the **<***object***>** to indicate end-of-range. Otherwise, the text in the range is deleted and you're placed in input mode. When within a line, the specified range of text isn't deleted until you type Enter.
[*n*]**cc**	Change *n* lines.
R	Replace the rest of the line with text. Rather than delete the text as with the change command, before going into insert mode, all text entered overwrites the current line until it has been completely replaced; at that point you're placed in input mode.
s	Substitute text for the current character (abbreviation for **cl**).
S	Substitute text for the current line (abbreviation for **cc**).

Input mode

In input mode, all keystrokes are inserted into the text at the cursor's position, except for the following:

Esc *or* Ctrl – [	Exit from input mode, back to command mode.
INTR	Interrupt execution (usually Ctrl – C, see **stty**).

erase *or* Ctrl – H	Erase the character before the cursor.
Ctrl – W	Erase the last input word.
Ctrl – A	Insert a copy of the last input text.
Ctrl – D	Delete one indent character.
Ctrl – L	Redraw the screen.
Ctrl – M *or* Enter	Insert a **newline**.
Ctrl – P	Insert the contents of the cut buffer.
Ctrl – R	Redraw the screen (like Ctrl – L).
Ctrl – T	Insert an indent character.
Ctrl – U	Backspace to the beginning of the line.
Ctrl – V	Insert the following keystroke, even if special (e.g. Ctrl – V Ctrl – L would insert a **form-feed**).

4. Deleting, yanking, putting

As mentioned in the Introduction, **elvis** supports very powerful operator constructs. In this section we'll deal with the delete, yank, and put operators.

delete	Delete the specified region and place the text into the unnamed buffer or a named buffer, if one is specified.
yank	Make a copy of the specified region and place it into the unnamed buffer or a named buffer, if one is specified.
put	Put the contents of the unnamed buffer, or of the named buffer, if specified, into its new location.

© 2005, QNX Software Systems

When you have a sequence of text you want to copy or move to a different location, you would first use the yank or delete operators to copy or delete the data into a buffer; you would then specify the **put** command to place the data at its new location. You can switch to another file before putting the text back, thus you can copy from one file to another.

Copying

To copy the next four lines, you would yank them with a command such as:

```
4yy    or    y4j
```

then you would move to a new location and put the text by typing:

```
p
```

Deleting

To delete the next four lines, you would use a command such as:

```
4dd    or    d4j
```

Moving

To move the next four lines, you would use a command such as:

```
4dd    or    d4j
```

then you would move to a new location and put the text by typing:

```
p      or    :pu
```

5. Filters

With the filter command, you can select regions of text and run them through any command and insert the output into the file. The text in the range specified from the current line to the delimited **<***object***>** is filtered through the command and replaces the region specified. The text that was in the region before being replaced by the output of the command is saved in the unnamed buffer — or in a named buffer if one was specified. If uppercase letters are used, **elvis** will append text to the named buffer.

!<*object***>***command*

Delete the specified text object into a buffer (unnamed, or named if given). Pass the specified text region to the standard input of the command, and replace it with the command's output. When you enter this command, the **!** prompt won't appear until an **<***object***>** has been given.

For example, to sort the next 10 lines:

```
!10!sort
```

6. Shifting text

The shift operators, **<** and **>**, shift all the lines delimited by the current line and the **<***object***>** operand. Text is shifted by the value of the **shiftwidth** option (see below). The forms of the shift command are:

><*object***>**	Shift **<***object***>** to the right.
<<*object***>**	Shift **<***object***>** to the left.
[*n*]**>>**	Shift next *n* lines to the right.
[*n*]**<<**	Shift next *n* lines to the left.

© 2005, QNX Software Systems

☞ "Named Buffer" prefixes don't work with the shift operator.

7. Miscellaneous commands

[*n*]**.**	Repeat last text modifying command *n* times.
[*n*]**r<***char***>**	Replace current character with *char*.
D	Delete to the end of the line (abbreviation for **d$**)
[*n*]**J**	Join the next line with the current line.
:[*x*,*y*]**j**	Join all the lines in the specified range.
[*n*]**x**	Delete *n* chars, to the right, including cursor position.
[*n*]**X**	Delete *n* chars to the left of the cursor.
[*n*]**~**	Reverse the case of the *n* next characters.
:[*x*,*y*]**p**	Display text in the specified range.
:[*x*,*y*]**l**	Display text in the specified range with tab and end-of-line markers.
:[*x*,*y*]**nu**	Display text in the specified range, with line numbers.
:so *file*	Read and execute the commands listed in *file*.
:ve	Display version number and compilation date of **elvis**.

8. Searching

Patterns

A pattern used for searching and substituting is called a *regular expression*. While most characters will match themselves in a search request, some have special meaning. To use these special characters in a search expression, put a backslash (\) before them:

Character	Meaning
^	beginning of line
$	end of line
.	any single character except the **newline**
\<	the beginning of a word
\>	the end of a word
[*string***]**	any single character in *string*
[^*string***]**	any single character not in *string*
[*x***-***y***]**	any character between *x* and *y* (range)
[^*x***-***y***]**	any character not between *x* and *y* (range)
*****	zero or more consecutive occurrences of the previous character
****	Escape special characters

See the **grep** utility for more information on regular expressions.

Search commands

/[*pattern***]**Enter

Search forward for *pattern*.

?[*pattern***]**Enter

Search backward for *pattern*.

/[*pattern***]/{+|-}***n*Enter

Go to the *n*th line relative to the line on which *pattern* is found, in forward direction.

?[*pattern***]?{+|-}***n*Enter

Go to the *n*th line relative to the line on which *pattern* is found, in backward direction.

/Enter — Repeat the previous pattern search, in forward direction.

?Enter — Repeat the previous pattern search, in backward direction.

n — Repeat the previous pattern search.

N — Repeat the previous pattern search, in reverse direction.

Search examples

/fred[1-3]	Match **fred1**, **fred2**, **fred3**
/^QNX	Match **QNX** at the beginning of a line
/QNX$	Match **QNX** at the end of a line

9. Global & substitute commands

Substitute command

The substitute command substitutes text matching a pattern with replacement text:

:[*x*,*y*]**s/***pattern***/***replacement text***/**[**c**][**g**][**p**]

If none of the modifiers **c**, **g**, or **p** are specified, this command will replace the first occurrence of the given pattern. You can modify this behavior by specifying any combination of the three modifiers:

c prompt before replacing

g replace all matched occurrences on a line

p display all lines containing the replaced text

This command is very powerful when used in conjunction with the global command (see below).

Ranges

You can specify a range wherever **[***x***,***y***]** is indicated. The first element, *x*, indicates the first line of the range while the second element, *y*, indicates the last line. Range elements can be line numbers or any of the following special characters:

. the current line

$ the last line in the file

% same as **1,$** (i.e. the entire file)

n relative offset from current line

A range element may also be a pattern specification:

/pattern/

or a marked location:

'<a-z>

For example, to print the lines from the next line containing “steve” until the first subsequent blank line:

```
/steve,/^$/!lp
```

© 2005, QNX Software Systems

Global command

The global command searches through the lines of the specified range-or through the whole file if no range ([*x*,*y*]) is specified-for lines that contain the pattern. For each matching line, *command* will be performed on it. The global command is of the following form:

:[*x*,*y*]**g/***pattern***/***command*

Execute *command* for every line that matches *pattern*.

:[*x*,*y*]**g!/***pattern***/***command*
:[*x*,*y*]**v/***pattern***/***command*

Execute *command* for every line that doesn't match *pattern*.

You can combine the substitute and global commands, using the following syntax:

[*x*,*y*]**g/***pattern***/s//***replacement_text***/**

This command runs the substitute command on every occurrence of a matched pattern within the given range. The null pattern specification (i.e. **//**) indicates to the substitute command that it is to use the currently matched global pattern as the text it is to replace.

The following variations may also be used:

[*x*,*y*]**g!/***pattern***/s//***replacement_text***/**
[*x*,*y*]**v/***pattern***/s//***replacement_text***/**

Global examples

Substitute the word **fred** with the word **barney** from lines 1 to 10:

```
1,10g/fred/s//barney
```

Match every **mary** that's at the beginning of a line and prompt the user to confirm the substitution:

```
1,$g/^mary/s//dave/c
```

10. Undo and retrieving

Undo

On occasion, you need to undo the effects of a command.

u Undo the effects of the last command that changed the edit buffer.

U Undo the effects of all the text modifying commands performed on this line. Return the line to its original state.

Retrieving previous deletions

The editor saves the last nine deleted blocks of text in save buffers. You can retrieve a buffer with the following command:

"*n***p** Retrieve *n*th previous deletion (1-9). Place after the cursor.

"*n***P** Retrieve *n*th previous deletion (1-9). Place before the cursor.

If you accidentally select the wrong buffer, you can use the undo command to clear it and then try specifying a different buffer.

You can also use named buffers (See “Deleting, yanking, putting”):

"<a-z>p Retrieve the contents of the named buffer. Place after the cursor.

"<a-z>P Retrieve the contents of the named buffer. Place before the cursor.

© 2005, QNX Software Systems

11. Screen commands

Scrolling

[*n*]Ctrl – D — Scroll down *n* lines (default is half page). *n* is remembered and becomes the default.

[*n*]Ctrl – U — Scroll up *n* lines (default is half page). *n* is remembered and becomes the default.

[*n*]PgDn *or*
[*n*]Ctrl – F — Jump forward *n* pages.

[*n*]PgUp
[*n*]Ctrl – B — Jump backward *n* pages.

[*n*]Ctrl – E — Scroll down *n* lines.

[*n*]Ctrl – Y — Scroll up *n* lines.

Screen adjustments

To position the current line at different positions on the screen by scrolling forward/backward:

z. — Place line at center of screen.

z- — Place line at bottom of screen.

Ctrl – L
Ctrl – R

Redraw the screen.

Status

Ctrl – G *or*
:f — Display status line on bottom of screen.

12. File manipulation

Writing to a file

All **w** commands can be preceded by an [*x*,*y*] range. For example, to write the first 6 lines you could specify: **1,6w**

:w	Write changes to current file.
:w *file*	Write changes to *file*.
:w >>*file*	Append changes to *file*.
:w! *file*	Force write to *file*.
:wq	Write the file and exit.
ZZ *or* **:x**	Exit **elvis**. If any changes were made, the edit buffer is written to the file.

Editing other files

:e *file*	Edit *file*.
:e!	Re-edit current file, discard changes.
:e! *file*	Edit *file*, discard changes to current file.
:e +*n file*	Edit *file*, start at line *n*.
:e # *or* Ctrl – ^	Return to the previous position in the last edited file.
:n	Edit the next file in the argument list.
:n!	Edit the next file, discard changes.
:n *args*	Use this new argument list.
:args	Show list of files, **[]** indicate current file.
:rew	Rewind list of files, re-edit first file.

© 2005, QNX Software Systems

:rew! Rewind the list of files, re-edit the first file, and discard any changes to the current file.

(See the **:ta**gs movement command)

Reading in a file

:r *file* Read in *file* after the cursor.

:*n***r** *file* Read in *file* after line *n*.

:r !*cmd* Read in the output of the named command.

Leaving **elvis**

ZZ *or* **:x** Exit **elvis**. If any changes were made, the edit buffer is written to the file.

:wq Write the file and exit.

:q Exit from **elvis**.

:q! Exit from **elvis** and discard changes.

Q Escape to the **ex** line editor (to return, type **elvis**).

13. Escaping to a shell

To execute a single command from within **elvis**, you can use the following command:

:!*cmd* Execute a single command, then return to **vi**.

Sometimes you may wish to execute more than one command, so you would probably want to create a shell.

:sh Start a subshell. You can type Ctrl – D or **exit** to return to **elvis**.

14. Abbreviations and macros

Macros

Macros let you bind a single key to an arbitrary set of editing commands.

Defining a macro is simple: you define an *lhs* (left-hand side), which is the single character you want translated, followed by an *rhs* (right hand side), which is the sequence it will be mapped into.

:map *lhs rhs*	Create macro
:unmap *lhs*	Delete macro.
:map	Display macros

For example:

```
:map q :wq<Ctrl – V> <Enter> <Enter>
```

This would map **q** into **:wq<**Enter**>**. The Ctrl – V is needed to quote (escape) the first Enter, while the second Enter ends the **map** definition.

Abbreviations

Word abbreviations are similar to macros, but they expand a short word into a longer word or words. If you type the word as part of a longer word, it's left alone. Abbreviations are used in input mode primarily to save typing.

Substitution isn't performed until you type a non-alphanumeric character to mark the end of the word. If you type Ctrl – V before the non-alphanumeric character, **elvis** won't perform the substitution.

:ab *abbr replacement_text*

Create an abbreviation.

:ab	Display abbreviations.

© 2005, QNX Software Systems

:una *abbr* Turn off this abbreviation.

For example:

```
:ab woof mary had a little ram
```

This would insert the text "**mary had a little ram**" whenever the word "**woof**" is typed.

15. Options

You can set or examine options via the colon command **set**. The values of options will affect the operation of subsequent commands.

There are three option types:

- those that toggle on or off; these are enabled by typing **:set** *option* and disabled by typing **:set no***option*.
- those that expect a numeric value; these are used by typing **:set** *option***=***value*
- those that expect a string value; these are used by typing **:set** *option***=***string*

To print all the option settings, type:

```
:set all
```

For convenience, options have both a long descriptive name and a short name that's easier to type. You may use either interchangeably.

Long name	Short	Default	Meaning
autoindent	ai	noai	Auto-indent during input
autoprint	ap	ap	In **ex**, print the current line

continued...

Long name	Short	Default	Meaning
autowrite	aw	noaw	Auto-write when switching files
directory	dir	dir="/tmp"	Where temporary files are kept
edcompatible	ed	noed	Remember `:s//` options
errorbells	eb	eb	Ring bell on error
ignorecase	ic	noic	Upper/lowercase distinction ignored in search
list	li	noli	Display lines in "list" mode
magic	ma	ma	Allow regular expression in search
paragraphs	pa	pa="PPppIPLPQP"	Names of "paragraph" for text formatter
readonly	ro	noro	Prevent overwriting of original file
report	re	re=5	Report when 5 or more changes
scroll	sc	sc=12	Scroll amount for Ctrl – U and Ctrl – D
sections	se	se="NHSHSSSEse"	Names of "section" for text formatter

continued...

© 2005, QNX Software Systems

Long name	Short	Default	Meaning
shell	sh	sh="/bin/sh"	Full pathname of the shell
showmatch	sm	nosm	Show matching ( ) [ ] **{ }**
showmode	smd	nosmd	Say when we're in input mode
shiftwidth	sw	sw=8	Shift amount for **<** and **>**
tabstop	ts	ts=8	Width of **tab** characters
term	te	te="$TERM"	Name of terminal
warn	wa	wa	Warn for ! if file is modified
wrapmargin	wm	wm=0	Wrap long lines in input mode
wrapscan	ws	ws	At EOF, searches wrap to line 1

You can configure the option settings with the **EXINIT** environment variable:

```
EXINIT="set ai aw"
export EXINIT
```

AutoIndent If you specify **autoindent** during input mode, each added line will begin with the same amount of leading white space as the line above it. Without **autoindent**, added lines begin at column zero.

AutoPrint This option affects only **ex** mode. If this option is on, and either the cursor has moved to a different

line or the previous command modified the file, **elvis** will print the current line.

AutoWrite If you've made modifications to the current file then try switching to another file, (e.g. **:tag**, or **:next**...), **elvis** will normally print an error message and refuse to switch. If this option is on, **elvis** will write the modified version of the current file and successfully switch to the new file.

DIRectory **elvis** stores text in temporary files. This option lets you control which directory those temporary files will appear in. The default is **/tmp**.

You can set this option only in a **.exrc** file; after that, **elvis** will have already started making temporary files in some other directory, so it would be too late.

If your **/tmp** directory is on ramdisk, recovery won't be possible if the system is rebooted.

ErrorBells The **elvis** utility normally rings a bell when you do something wrong. This option lets you disable the bell.

IgnoreCase Normally, when **elvis** searches for text, it distinguishes between uppercase and lowercase letters. When this option is on, uppercase and lowercase are seen as equivalent.

LIst In **nolist** mode (the default), **elvis** displays text in a "normal" manner — with **tab**s expanded to an appropriate number of spaces, etc. However, sometimes it's useful to have **tab** characters displayed differently. In list mode, each **tab** is displayed as **^I**, and a $ is displayed at the end of each line.

© 2005, QNX Software Systems

MAgic The search mechanism in **elvis** can accept *regular expressions* — strings in which certain characters have special meaning. The **magic** option is normally on, which causes these characters to be treated specially. If you turn the **magic** option off (**:set noma**), then all characters except ^ and $ are treated literally. Both ^ and $ retain their special meanings regardless of the setting of **magic**.

PAragraphs The **{** and **}** commands move the cursor forward or backward in increments of one paragraph. Paragraphs may be separated by blank lines or by a leading "dot" command of a text formatter. Different text formatters use different dot commands. This option lets you configure **elvis** to work with your text formatter.

It's assumed your formatter uses commands that start with a "**.**" character at the front of a line, followed by a one- or two-character command name.

The value of the **paragraphs** option is a string in which each pair of characters is one possible form of your text formatter's paragraph command. For example, lines starting with **.IP**, **.LP**, or **.PP** would be considered new paragraphs.

ReadOnly Normally, **elvis** lets you write back any file for which you have write permission. If you don't have write permission, all you can do is write the changed version of the file to a different file.

If you set the readonly option, **elvis** pretends you don't have write permission to any file you edit. This is useful when you mean to use **elvis** only to look at a file, not to change it. This way, you can't change the file accidentally.

This option is normally off, unless you use the **view** alias of **elvis**, which is like **elvis** except the **readonly** option is on.

REport Changes, deletes, and yanks may affect many lines. For commands affecting a lot of lines, **elvis** outputs a message saying what was done and how many lines were affected. This option lets you define what "a lot of lines" means. The default is 5, so a message will be shown on the status line for any command affecting 5 or more lines.

SCroll The Ctrl – U and Ctrl – D commands normally scroll backward or forward by half a screenful, but you can adjust this. The value of this option says how many lines those keys should scroll. You can also set this with [*n*]Ctrl – D and [*n*]Ctrl – U.

SEctions The **[[** and **]]** commands move the cursor backward or forward in increments of one section. Sections may be delimited by a **{** character in column 1 (which is useful for C source code) or by means of a text formatter's "dot" commands.

This option lets you configure **elvis** to work with your text formatter's **section** command, in exactly the same way that the paragraphs option makes it work with the formatter's **paragraphs** command. For example, lines starting with **.SH** or **.NH** would be considered new sections.

SHell When **elvis** forks a shell (perhaps for the **:!** or **:sh** commands), this is the program it uses as a shell. The normal default is **/bin/sh**. However, if you have set the **SHELL** environment variable, the default value is copied from the environment.

ShiftWidth The **<** and **>** commands shift text left or right by a uniform number of columns. The **shiftwidth**

© 2005, QNX Software Systems

option defines that uniform number. The default is 8 spaces.

ShowMatch With showmatch set, every time you type), **}**, or] in input mode, **elvis** will momentarily move the cursor to the matching character:

```
)     --->     (     --->     )
}     --->     {     --->     }
]     --->     [     --->     ]
```

ShowMoDe In visual mode, it's easy to forget whether you're in the visual command mode, or input/replace mode. Normally, the **showmode** option is off, and you aren't informed as to which mode you're in. If you turn the **showmode** option on, a message will appear in the lower right corner of your screen, telling you which mode you're in.

TabStop Normally, **tab** characters are 8 characters wide, but you can change their width with this option.

TErm This "read-only" option shows the name of the terminal entry that **elvis** is using for your terminal.

WArn If you've modified a file, but not yet written it back to disk, **elvis** will normally print a warning before executing a **:!***cmd***"** command. However, in **nowarn** mode, this warning isn't given.

Normally, **elvis** also prints a message after a successful search that wrapped at EOF. The [**no**]**warn** option also disables this warning.

WrapMargin Normally (with **wrapmargin=0**), **elvis** lets you type in extremely long lines.

However, with **wrapmargin** set to something other that 0 (**wrapmargin=70** is nice), long lines are

automatically “wrapped” on a word break for lines longer than **wrapmargin**’s setting.

WrapScan Normally, when you search for something, **elvis** will find it no matter where it is in the file. **elvis** starts at the cursor position, and searches forward. If **elvis** hits EOF without finding what you’re looking for, then it wraps around to continue searching from line 1, up to the current line.

If you turn off the **wrapscan** option (**:se nows**), and **elvis** subsequently hits EOF during a search, it will stop and say so.

16. ex commands

The following list of **ex** commands is intended for your convenience only. It’s beyond the scope of this document to describe the operations of these commands in detail. Most of these commands have been indicated as alternatives to the visual mode commands described in the above sections.

The general form of these commands is:

:[*x*,*y*] *command parameter*

[Prefix]	Command	Short form
	abbrev *abbr text*	**ab**
[*line*]	**append**	**a**
	args	**ar**
[*x*[,*y*]]	**change**	**c**
[*x*[,*y*]]	**copy** *line*	**co**
[*x*[,*y*]]	**delete** [**"***named_buffer*]	**d**

continued...

© 2005, QNX Software Systems

[Prefix]	Command	Short form
	edit[**!**] *file*	**e**
	file	**f**
[*x*[**,***y*]]	**global /***pattern***/***command*	**g**
[*line*]	**insert**	**i**
[*x*[**,***y*]]	**join**	**j**
[*x*[**,***y*]]	**list**	**l**
	map *lhs rhs*	**map**
[*x*[**,***y*]]	**move** *line*	**m**
	next[**!**]	**n**
[*x*[**,***y*]]	**number**	**nu**
[*x*[**,***y*]]	**print**	**p**
[*line*]	**put** [**"***named_buffer*]	**pu**
	quit[**!**]	**q**
[*line*]	**read** *file* **| !***command*	**r**
	rewind[**!**]	**rew**
	set [*option*]	**se**
	shell	**sh**
	source *file*	**so**
[*x*[**,***y*]]	**substitute /***pattern***/***text***/**[**c**][**g**][**p**]	**s**
	tag	**ta**
	unabbrev *abbr*	**una**
	undo	**u**
	unmap *lhs*	**unm**

continued...

[Prefix]	Command	Short form
	version	**ve**
	visual	**vi**
[*x*[**,***y*]]	**write**[**!**][[**>>**]*file*]	**w**
	xit[**!**]	**x**
[*x*[**,***y*]]	**yank** [**"***named_buffer*]	**ya**

The following **ex** commands have no long version:

Command	Form
escape	**!***command*
print next	**CR**
lshift	**<**
rshift	**>**

Files:

/tmp/elv*
: During editing, text is stored in a temporary file in **/tmp**.

tags
: Tag database created by **ctags** and used by the **:tags** command and the **-t** option.

$HOME/.exrc
: On startup, the contents of this file are executed as a series of **ex** commands.

Environment variables:

EXINIT
: If set, the contents of this environment variable will be executed on startup as a series of **ex** commands.

© 2005, QNX Software Systems

HOME Used to look for default configuration files and **.*rc** files.

LINES, **COLUMNS**

Overrides the screen size values associated with your terminal type.

SHELL Used to determine the command line interpreter to use for running commands from within **elvis**.

The **elvis** utility requires that the **TERM** environment variable be set to indicate your terminal type. For example, if **elvis** is running on a console, **TERM** should be set as follows:

```
export TERM=qansi
```

or

```
export TERM=qnx
```

depending on the mode your console is running in (QNX vs ANSI).

Contributing author:

Steve Kirkendall

Caveats:

When displayed, long lines scroll horizontally. On some implementations, these wrap onto multiple rows on the screen.

See also:

ctags, **elvrec**, **qed**, **sed**, **vedit**

Linda Lamb, *Learning the vi Editor*, O'Reilly and Associates, 1990

Syntax:

elvrec [*preservedfile* **[***newfile***]]**

Options:

preservedfile	The name of the file you had been editing with **elvis** that is to be recovered.
newfile	The name of the file you want the recovered text written to, instead of the original name of the preserved file.

Description:

If you're editing a file when **elvis** dies, the system crashes, or power fails, the most recent version of your text will be preserved. The preserved text is stored in a special directory; it does *not* overwrite your text file automatically.

The **elvrec** program locates the preserved version of a given file, and writes it over the top of your text file — or to a new file, if you prefer. The recovered file will have nearly all of your changes.

To see a list of all recoverable files, run **elvrec** with no arguments.

Examples:

Write a list of all recoverable files to the standard output:

```
elvrec
```

Recover the file **myfile.c** which you had been editing when the power failed:

```
elvrec myfile.c
```

Recover the file **stuff** and place the recovered text in the file **newstuff**:

```
elvrec stuff newstuff
```

© 2005, QNX Software Systems

Files:

/usr/preserve/p*

The text that was preserved when **elvis** died.

/usr/preserve/Index

A text file which lists the names of all preserved files, and the names of the **/usr/preserve/p*** files which contain their preserved text.

Author:

Steve Kirkendall

Caveats:

The **elvrec** utility is very picky about filenames. You must tell it to recover the file using exactly the same pathname as when you were editing it. The simplest way to do this is to go into the same directory that you were editing, and invoke **elvrec** with the same filename as **elvis**. If that doesn't work, then try running **elvrec** with no arguments, to see exactly which pathname it is using for the desired file.

Due to the permissions on the **/usr/preserve** directory, on UNIX systems **elvrec** must be run as **root**. This is accomplished by making the **elvrec** executable be owned by **root** and setting its "set userid" bit (**chmod u+s**).

If you're editing a nameless buffer when **elvis** dies, then **elvrec** will pretend that the file was named **foo**.

See also:

elvis

Syntax:

```
emu87 [-f] &
```

Options:

-f Force the emulator to remain, even if floating-point hardware is present.

Description:

The **emu87** starts the floating-point emulator which traps 80287 and 80387 floating-point instructions and emulates them through software. By default, the emulator will quietly terminate if you attempt to start it on a machine equipped with floating-point hardware. You can force the emulator to remain active by using the **-f** option.

The floating-point emulator is required if you wish to run programs that need floating-point support on a computer that doesn't have an 80X87 floating-point coprocessor.

The **emu87** utility is not the emulator itself. Instead, it starts the following emulators:

emu387 an 80387 emulator for 32-bit programs

emu87_32 *or* **emu87_16**

an 80287 emulator for 16-bit programs. When a 32-bit process manager (**Proc**) is running, **emu87_32** will be started. If an older 16-bit version of **Proc** is running (QNX 4.21 or earlier), **emu87_16** will be started.

To remove an emulator, use the **slay** utility,

If you are running on a processor without hardware floating-point, and do wish to be able to run 16-bit floating-point programs, you may invoke **emu387** directly thus avoiding unnecessarily starting **emu87_32**.

© 2005, QNX Software Systems

Never attempt to start or stop the emulator while processes are actively using floating-point, or those processes will fault on their next use of floating-point.

Examples:

Start the floating-point emulator:

```
emu87 &
```

Files:

The **emu87** utility may use several 80x87 emulators which are found in **/bin**:

- **/bin/emu387**
- **/bin/emu87_32**
- **/bin/emu87_16**

Syntax:

env [-i] [*name***=***value***]... [***command* **[***arguments***]]**

Deprecated:

env - [*name***=***value***]... [***command* **[***arguments***]]**

Options:

-i	Ignore the environment that would otherwise be inherited from the current shell. This restricts the environment for *command* to that specified by any *name***=***value* pairs.
-	(Deprecated) same as **-i**
*name***=***value*	Set the environment variable filename to *value* and add it to the environment.
command	A command to be invoked.

Description:

The **env** utility obtains the current environment, modifies it according to its arguments, and executes *command* with the modified environment. If no command is specified, the resulting environment is displayed.

If you have removed the **PATH** environment variable from the environment, you must include the path to the command.

Examples:

Start a shell with only the **SHELL** and **PATH** environment variables:

```
env -i SHELL=/bin/sh PATH=/bin:/usr/bin sh
```

Start a daemon process with no environment:

```
env -i /bin/cron
```

Exit status:

When a *command* is specified on the command line, the **env** utility will attempt to *exec()* into that command. If the command is successfully launched, the exit status from **env** will be that of the program it *exec()*ed into. Otherwise, **env** will return an exit status as follows:

0 (Assuming no command was specified) the resulting environment was successfully displayed.

>0 An error was present in the command-line parameters supplied or **env** was unable to launch the command specified on the command line.

Syntax:

errno *error_number*...

Options:

error_number	Get the text equivalent of this numerical error number.

Description:

The **errno** utility will display the string equivalent for the *error_number*s supplied on the command line. The output is written to the standard output.

This utility is useful in cases where a program has indicated that a numerical error occurred and has not provided the ASCII string equivalent of that *errno* value. Users who do not have access to the C header file **/usr/include/errno.h** do not have the option of using "**fgrep** *error_number* **/usr/include/errno.h**". The **errno** utility is more convenient and is available to all users.

Examples:

Find the string equivalent of error number 2:

```
$ errno 2
errno    2: No such file or directory
$
```

Files:

The **errno** utility writes one newline-terminated line to the standard output for each command-line argument it is passed. In those cases where the command-line argument is not a valid error number, a line is written to the standard error instead. The standard input is not used.

© 2005, QNX Software Systems

Exit status:

>0	An error occurred (e.g. unknown errno value)
0	The error string was successfully written to the standard output.

See also:

/etc/config/traceinfo file, **netinfo**, **traceinfo**, **use**

Syntax:

esh [-c *command***] [-irv] [***script_file***]**

Options:

-c *command*	Run this command.
-i	Enter interactive mode after running a script file. If you don't specify this option, **esh** will terminate after running a script.
-r	Run in restricted mode. In this mode, you can't perform certain operations. If you try to, you'll see the error message, "Operation not permitted." Restricted operations include creating remote processes and specifying a path that starts with a double slash for **cd** and other command names. For more information, see the sections "Command-line format" and "Built-in commands."
-v	Be verbose: echo each command before executing it.
script_file	A file containing shell commands to execute.

Description:

The **esh** utility provides a subset of the functionality found in the standard QNX Shell, **/bin/sh**. You should find **esh** useful for situations where memory requirements are limited. For example, you could use it to run a simple system initialization file for an embedded system.

Command-line format:

In **esh**, command lines take this form:

[**//***node*] *command arg1 arg2 ...* [*redir-op file*] [**&**]

© 2005, QNX Software Systems

Where:

//node — The node the command is to be executed on. Remote execution isn't allowed in restricted mode. See the section entitled "Remote execution" for the **sh** utility.

command — The command to be executed. If it doesn't start with a slash, the command will follow the path set by the **PATH** environment variable. In restricted mode, a command can't start with a slash.

redir-op file — A redirection operator. When a command is invoked, three standard files are set up in its environment. These files, *standard input*, *standard output*, and *standard error* output (*stdin*, *stdout*, *stderr*), are usually attached to the active terminal. You can redirect a command's standard input, standard output, and standard error as follows:

Specifying:	**Will:**
<*file*	redirect standard input from this file.
>*file*	redirect standard output to this file. If the file exists, it's overwritten; if the file doesn't exist, it's created.
>>*file*	redirect standard output to this file. If the file exists, the information is appended to the end of the file; if the file doesn't exist, it's created.
2>*file*	do the same as **>***file*, but for standard error.
2>>*file*	do the same as **>>***file*, but for standard error.

continued...

Specifying:	Will:
&	If a command contains an unquoted &, then **esh** doesn't wait for the command to complete execution but immediately moves on to process the next command. The standard input of the command is redirected from **/dev/null**, and SIGINT and SIGQUIT are ignored.

Filename expansion:

You use most QNX commands for manipulating files in one way or another. As such, **esh** has a filename "shorthand" (consisting of *****, **?**, **[**, and **]**) you can use to specify the files that a command is to operate on. This shorthand is the same used by the standard QNX Shell. For more information, see "Filename expansion" in **sh**.

Quoting:

The following characters have a special meaning in **esh**:

```
&  \  "  *  ?  [  space
```

To suppress the special meaning of these characters and keep their literal meaning, you use quoting.

To quote a sequence of characters or sequence of words, enclose the sequence in double quotes. To quote a single character, use double quotes or precede it with the escape character (\).

Escape character (backslash)

The escape character (\) preserves the literal meaning of the next character. You can't obtain a single backslash by quoting \ with double quotes. To obtain a backslash, enter \\ instead.

© 2005, QNX Software Systems

Double quotes

Enclosing characters and words in double quotes (**""**) preserves the literal meaning of all characters within double quotes, with the exception of the \ character. For example:

```
"ab cd"
```

represents a single, five-character argument.

You can keep the literal meaning of a double quote with the \ character. For example:

```
ab\"cd
```

represents the single, five-character argument **ab"cd**.

Builtin commands:

The following commands are built into **esh**. That is, **esh** interprets and executes them internally.

alias list aliases

alias [*name***=***value*]...

Without arguments, **alias** lists all aliases and their values. If only the name is specified, its value is listed. Any name specified with a value defines an alias.

Alias expansion occurs when the first word of a statement is a defined alias, except when that alias is already being expanded.

List all aliases:

```
alias
```

Remove an alias:

alias *name***=**

cd change working directory

cd [*directory*]

The **cd** utility changes the working directory of the current execution environment. If *directory* isn't specified, the value of the **HOME** environment variable becomes the new working directory.

. *file* execute commands in the current environment. The **.** (dot) command reads and executes the commands from *file* within the current environment. The search path contained in the environment variable **PATH** is used to find the directory containing *file*. This command doesn't require that *file* be executable.

exec execute a command and/or manipulate file descriptors

exec [*command* [*argument*...]]

The **exec** command opens, closes, or copies file descriptors as specified by any I/O redirections given as part of *argument*. If a command is specified, that command is spawned as a replacement for **esh**. Any specified arguments are passed to the spawned process.

If **esh** is operating in restricted mode (**-r**), the **exec** command cannot be used.

exit cause **esh** to exit

exit [*n*]

The **exit** command causes esh to exit with the exit status specified by *n*. If *n* isn't specified, **esh** exits with the status of the last command executed.

export mark environment variables for export

export *name*[**=***word*]...
export -p

The **export** command marks the specified names for **export**, which will cause them to be in the environment of subsequently executed commands. If

you specify the **-p** option, the names and values of all exported variables are written to the standard output.

If restricted mode (**-r**) is set, you can't use this command.

kill *pid* set a SIGTERM on the process with process ID *pid*.

reopen attach input and output to a device

reopen [*device*]

The **reopen** command closes standard input, standard output, and standard error and attaches them to the specified device. This command is often used in **sysinit** files. If you don't specify a device, **//0/dev/con1** is used.

If restricted mode (**-r**) is set, you can't use this command.

set -v Enable verbose mode.

This command causes all commands to be echoed to the standard output before they're executed.

Examples:

Invoke a subshell:

```
esh
```

Invoke a subshell with a script file:

```
esh /etc/backup
```

Run the following command and exit:

```
esh -c "ls /bin"
```

Files:

/etc/esh	ASCII text file containing shell commands. It is executed when **esh** is started as a login shell.

Environment variables:

You can use the following variables to modify **esh**'s environment:

HOME	The pathname of the user's home directory
LOGNAME	The user's **login** name.
PATH	The directory search path used by esh for locating executable programs and **esh** shell scripts. To change **PATH**, you must use the **export** command. If **PATH** isn't in the existing environment when **esh** is invoked, it is set to **/bin:/usr/bin**.
SHELL	The pathname of the user's preferred shell.
TERM	The terminal type.
TMPDIR	The pathname of a directory that can be used by utilities that need a place to create temporary files.
TZ	The timezone setting.

See also:

sh, **uesh**

expand

© 2005, QNX Software Systems

Convert tabs to spaces (POSIX)

Syntax:

expand [-t *tablist*] [*file*...]

Options:

-t *tablist* Set up tab stops according to the *tablist* argument. This argument consists either of a single positive decimal integer, or of multiple positive decimal integers, in ascending order, separated by single commas.

If no number is given, tabs stops are set eight columns apart. If a single number is given, tab stops are set *tablist* columns apart. If multiple numbers are given, tab stops are set at those columns.

file The pathname of a file whose tabs are to be converted.

Description:

The **expand** utility copies files or the standard input to the standard output with **tab** characters replaced by the number of **space** characters needed to pad to the next tab stop. Any **backspace** characters encountered in the input are copied to the output and cause the column position count for the tab-stop calculations to be decremented; the count is never decremented below zero.

The **-t** option lets you specify how many columns tab stops are set apart. You can also use it to specify a multiple tab-stop list that determines where tab stops are placed. If **tab** characters are present in the input past the last tab stop specified in a multiple tab-stop list, those **tab**s are each replaced by a single **space** in the output.

Note that tabbing to position *N* causes the next character written to appear in the *next* column position on that line (i.e. column *N*+1)

Examples:

For the file **myfile**, expand each **tab** to the number of **space**s required to reach the next tab stop:

```
expand myfile
```

Do the same as above, except set tab stops to every four columns rather than the default eight:

```
expand -t4 myfile
```

Place tab stops at the specified columns. Any **tab** encountered past the last tab stop will be replaced by a single **space**:

```
expand -t8,12,20,24,32,36,44,48 myfile
```

Files:

Input files are intended to be composed of ASCII text and white-space. Input is read either from the standard input or from one or more files named on the command line. Multiple input files are processed in the order they appear on the command line.

All output is written to the standard output.

If an error occurs (e.g. failure to open a file named on the command line), a diagnostic message is written to the standard error.

Exit status:

0	Successful completion.
>0	An error occurred.

© 2005, QNX Software Systems

Caveats:

The **expand** utility does not check to verify that the tabstops specified in the **-t** option are in ascending order, as is required for proper operation.

See also:

unexpand

Syntax:

expr *operand*...

Options:

operand Operand expression to evaluate. (See description)

Description:

The **expr** utility evaluates a single expression and writes the result to the standard output. The expression is formed from integer and string symbols in combination with the following operators:

```
(  )  |  &  =  >  >=  <  <=  !=  +  -  *  /  %  :
```

In the following table, expressions are listed in order of decreasing precedence, with equal-precedence operators grouped together. All of the operators are left-associative. Note that a *string* operand is an argument that cannot be identified as an **integer** argument or as one of the above operators.

Expression	Description
integer	An argument consisting only of an (optional) unary minus followed by digits.
string	A string argument.
(*expr* **)**	Grouping symbols. Any expression can be placed within parentheses.
expr1 **:** *expr2*	Matching expression.
expr1 ***** *expr2*	Integer multiplication.
expr1 **/** *expr2*	Integer division, producing an integer result.
expr1 **%** *expr2*	Remainder of integer division.

continued...

© 2005, QNX Software Systems

Expression	Description
expr1 **+** *expr2*	Integer addition.
expr1 **-** *expr2*	Integer subtraction.
expr1 **=** *expr2*	Equal.†
expr1 **>** *expr2*	Greater than.†
expr1 **>=** *expr2*	Greater than or equal.†
expr1 **<** *expr2*	Less than.†
expr1 **<=** *expr2*	Less than or equal.†
expr1 **!=** *expr2*	Not equal.†
expr1 **&** *expr2*	Returns the evaluation of *expr1* if neither expression evaluates to null or zero; otherwise, returns zero.
expr1 **|** *expr2*	Returns the evaluation of *expr1* if it is neither null nor zero; otherwise returns the evaluation of *expr2*.

† Returns the result of a decimal integer comparison if both arguments are **integer**s; otherwise, returns the result of a string comparison. The result of each comparison is 1 if the specified relation is TRUE or 0 if FALSE.

The "**:**" matching operator compares the string resulting from the evaluation of *expr1* with the regular expression pattern resulting from the evaluation of *expr2*. Usually, this operator returns the string representing the number of characters matched ("**0**" on failure). However, if the pattern contains at least one regular expression subexpression **\(**...**\)**, the string corresponding to **\1** is returned (see **grep**).

Examples:

Add 1 + 1 :

```
expr 1 + 1
```

Logical 1 **or** 0 :

```
expr 1 \| 0
```

Logical 1 **and** 0 :

```
expr 1 \& 0
```

Add 1 to $*a* :

```
expr $(expr $a + 1)
```

Count characters in $*var* :

```
expr "$var" : '.*'
```

Compare $*a* to a possible =

```
expr x$a = x=
```

Exit status:

0	The *expression* evaluates to neither null nor zero.
1	The *expression* evaluates to null or zero.
2	Invalid *expression*s were found.
>2	An error occurred.

Caveats:

The syntax for **expr** requires special attention. Many of the operators are also shell control operators or reserved words, so they have to be escaped on the command line. In addition, each part of the expression is composed of separate arguments, so you should use **blank**s liberally. For example:

© 2005, QNX Software Systems

Instead of entering:	You should enter:
`expr "1 + 2"`	`expr 1 + 2`
`expr 1 + (2 * 3)`	`expr 1 + \( 2 \* 3 \)`
`expr 1+2`	`expr 1 + 2`

In many cases, the arithmetic and string features provided as part of the shell command language are easier to use than their equivalents in **expr**. However, you may need **expr** to run older UNIX shell scripts.

When you want to specify the filesystem root, avoid the `/` character as a standalone argument — **expr** will interpret it as the division operator. Use `//` instead.

See also:

grep

false

Return false value (POSIX)

Syntax:

false

Options:

None.

Description:

The **false** utility does nothing and exits immediately with a non-zero exit code.

Exit status:

The **false** utility always exits with a value greater than zero.

See also:

true

Fatfsys

© 2005, QNX Software Systems

DOS FAT12/16/32 filesystem manager (QNX)

Syntax:

Fatfsys [*options***]... [***dos_drive***=***host_device***]... &**

Options:

-d Name of the directory to hold the DOS drives. (Default: **/dos**)

-D Enable the FAT32 volume to be marked "dirty" at mount, "clean" at unmount. This forces Windows to do a **ScanDisk** the next time it boots. (Default: disabled)

-f Set the maximum number of concurrent open files. This option also sizes the internal caches based on this value. (Default: 128)

-F Disable usage/update of the FAT32 **FSInfo** record, which contains the free cluster count and a hint of the next free cluster. By default **Fatfsys** loads and uses this information at mount, and writes back any changes at unmount. This option disables that usage.

-H Specify how to display DOS hidden files:

ignore	Never display hidden files.
show	Always display hidden files.
dot	Display hidden files with a leading "**.**" (useful with **ls -a**).

The default is **show**.

-i Set the user/group IDs to apply as the owner of all DOS files. These may be specified as numeric values (in the form of *num***:***num*) or as a user ID from **/etc/passwd** or group ID from **etc/group**. (Default: **0:0**)

-L Specify how to treat long filenames:

ignore	Never show or create long filenames; use only the corresponding short name (e.g. **PROGRA~1**).
show	Display and create long filenames where appropriate (i.e. if the filename is longer than 8.3 or is mixed-case).
always	Always create long filenames to preserve case.

The default is **show**.

-m Set mount options to apply to all subsequent DOS filesystem mountpoints. Valid options are:

ro	Mark file as read-only.
rw	Mark file as read-write.
noatime	Don't record the time the file was last accessed.
atime	Record the access time.
noexec	Don't allow executable files to load.
exec	Allow executable files to load.
nocreat	Don't allow files to be created.
creat	Allow files to be created.
async	Enable asynchronous writes.
sync	Enable synchronous writes.

The defaults are **rw**, **noatime**, **exec**, **creat**, **async**.

-P Set POSIX-like emulation/checking level. Valid options are:

ignore	Don't do any emulation or checks.

emulate Attempt to provide features not present in the DOS filesystem (e.g. link counts on directories, **.** and **..** entries at the root).

strict Use the **emulate** features, but also check for any attribute changes that DOS can't handle (e.g. a *chown()* to a user other than that specified in **-i**, or a *utime()* to a time before 1980, the DOS epoch). If you don't specify **strict**, such attribute changes are silently ignored.

The default is **ignore**.

-s Toggle the automatic device scanning sequence (based on user-specified mounts).

If any user mounts are specified on the command line, then **Fatfsys** won't scan the **/dev** directory. Use **-s** to toggle scanning on.

If no user mounts are specified on the command line, then **Fatfsys** will scan the **/dev** directory. Use **-s** to toggle scanning off.

-S Specify how to treat short (8.3) filenames (assuming **-L ignore** is specified):

lower Use all lowercase.

upper Use all uppercase.

windows Try to behave like Windows — use a leading uppercase letter and the rest lowercase, unless overridden by the case attribute in the directory entry.

The default is **lower**.

-t Set the timeout for removable media. After the timeout, the media is assumed to be possibly stale and is auto-remounted. (Default: 2 seconds)

-T Disable silent 8.3 name truncation. This option is valid only if long filenames are being ignored (via **-L ignore**). By default, DOS truncates filenames to fit into the 8.3 format, so the files **FILENAME.C** and **FILENAMETWO.C** are considered identical. The **-T** option will raise an ENAMETOOLONG error rather than truncate.

-u Set a umask to apply to all DOS files. A umask specifies a set of permission bits that should be cleared for a file. You can specify the umask using a numeric form or a symbolic string (of the form **u=rwx,g=rwx,o=rwx**) as an inverse umask. (Default: 0)

-v Increase level of verbosity for information/error messages.

-V Specify how to treat the DOS volume label:

- **ignore** Never display the volume label.
- **show** Show the label unchanged.
- **equals** Display the label with a leading **=** in the name.

When not being ignored, the volume label (if present) will be a name-special file with no access permissions.

The default is **equals**.

-w Specify how long to wait for devices to be set up. This is useful in **sysinit** or boot files. (Default: 10 seconds)

-X Control the setting of the executable bit:

- **none** No files will be executable.
- **all** All files will be executable.

system Use the DOS "system" attribute to indicate an executable file.

auto Let the DOS filename extension (e.g. ***.EXE**, ***.COM**, ***.BAT**) dictate if a file should be executable.

The default is **auto**.

dos_drive A DOS drive name (typically a letter from **a** to **z**).

host_device The name of the QNX device or file to open (e.g. **/dev/hd1t6**).

Description:

Fatfsys provides transparent access to DOS disks, so you can treat DOS filesystems as though they were QNX filesystems.

If started with no arguments, **Fatfsys** will scan the **/dev** directory for DOS devices to adopt. Hard drive partitions of types 1, 4, 6, 11, 12, and 14 will be assumed to contain a DOS filesystem. And any removable device (floppy, LS-120, etc.) will be assumed to be capable of having media containing a DOS filesystem inserted into it.

You may also specify explicit mount devices on the command line, using the form *dos_drive*=*host_device* (e.g. **c=/dev/hd1t6**). Specifying any device in this way will suppress the default device scan, but you can re-enable scanning by using the **-s** option.

Using mount and mount_dos

You can also mount DOS filesystems at runtime using the **mount** utility:

```
mount -t dos /dev/fd0 /dos/a
```

or the **mount_dos** utility:

```
mount_dos /dev/fd0 /dos/a
```

Typically, **mount_dos** is invoked via the **mount** utility. But if this isn't possible (e.g. if the device is being mounted from within the boot image), then you may run the **mount_dos** utility directly.

Using umount

You can unmount a DOS filesystem at runtime using the standard **umount** utility (e.g. **umount /dos/c**). Note that when **Fatfsys** is terminated (with a SIGTERM or simple **slay Fatfsys**), all current DOS filesystems are implicitly unmounted (this will affect FAT32 volumes; see the **-D** and **-F** options).

Removable media

Fatfsys maintains an activity timeout on removable media. After 2 seconds (see **-t**) of inactivity, the next attempted access to that filesystem will force a relearn of the DOS filesystem.

Fatfsys relies on the volume-tracking details of a disk to detect changes, then uses a heuristic (any open valid FDs) to determine if the media represents the same or a different filesystem.

Caching

Fatfsys maintains no local physical data block cache. All reads and writes to the host device are passed immediately to **Fsys**, which has a data cache.

To force **Fsys** to flush its cache, you may use the *fsync()* routine or **O_SYNC** flag on a per-FD basis, or use the **-m sync** option to affect all files written to the host device.

Fatfsys does contain a number of logical data caches (names, attributes, and cluster mappings), which are invalidated for removable media as described above. The size of these caches is based on the **-f** value.

Examples:

Start **Fatfsys**, mounting the floppy drive as **/dos/a**:

```
Fatfsys a=/dev/fd0 &
```

Start **Fatfsys**, mounting the floppy drive as read-only and the hard disk as read-write:

```
Fatfsys -m ro a=/dev/fd0 -m rw c=/dev/hd1t6 &
```

See also:

Blkfsys, **fdisk**, **Fsys**, **Fsys.***, **mount**, **mount_dos**, **textto**, **tr**, **umask**, **umount**

QNX *Installation & Configuration*

Display compressed (frozen) files (UNIX)

Syntax:

fcat [*filename***...]**

Options:

See **freeze**.

Description:

The **fcat** command is equivalent to **freeze -cd**. See the **freeze** utility for details.

The **freeze**/**melt**/**fcat** compression utilities will eventually become deprecated in favor of the GNU **zip** suite, **gzip**/**gunzip**/**zcat**. The **freeze** suite of utilities will continue to be provided for quite some time before being eliminated completely.

Author:

Leonid A. Broukhis

fdformat

© 2005, QNX Software Systems

Format floppy diskettes (QNX)

Syntax:

fdformat [-aiIpqv] [-h *heads***] [-t** *tracks***] [-n** *sectors***]**
[-s *size***] [-f** *first_track***] [-l** *last_track***]**
[-k *skew_factor***] [-z** *interlv***]** *drive*

Options:

-a	Abort on the first error encountered.
-f *first_track*	Start formatting at track *first_track*.
-h *heads*	The number of heads.
-i	Write layout information to the diskette, but don't format it.
-I	Ignore errors. Do not try to verify diskette.
-k *skew_factor*	The number of sectors by which **fdformat** should offset the starting sector of a track from the last sector of the previous track.
-l *last_track*	("el") Stop formatting at track *last_track*.
-n *sectors*	The number of sectors per track.
-p	Prompt before starting.
-q	Be quiet; don't report progress as each track is formatted.
-s *size*	The disk media: 360K, 720K, 1.2M, 1.4M, or 2.8M (overrides **-h**, **-t**, and **-n**).
-t *tracks*	The number of tracks.
-v	Be verbose; display extra information as the format progresses.
-z *interlv*	The interleave amount.
drive	The name of the physical device to be formatted (i.e. **/dev/fd0**).

Description:

The **fdformat** utility formats the specified disk. Formatting refers to a process of placing addressing marks and other control information on the disk to allow the hardware to read it. Formatting does not imply any sort of filesystem structure.

By default, **fdformat** uses the current mount characteristics of the drive as the drive parameters. These parameters may be overridden by specific command-line options, such as **-t** *tracks*.

The **-s** *size* option lets you specify the size of standard floppy disks, as in the following:

Size	Heads	Tracks	Sectors	Drive type
360K	2	40	9	$5\frac{1}{4}''$ floppy
720K	2	80	9	$3\frac{1}{2}''$ floppy
1.2M	2	80	15	$5\frac{1}{4}''$ floppy
1.44M	2	80	18	$3\frac{1}{2}''$ floppy
2.88M	2	80	36	$3\frac{1}{2}''$ floppy

QNX may mark disks with a volume label. This label allows easy identification of a disk by software (for more information, see **dinit**). By default, **fdformat** will refuse to format a disk if it encounters a volume label.

The first sector of QNX floppy disks contains some information about the layout of the diskette (heads, tracks, sectors). The **-i** option simply writes this layout information to a diskette without formatting it.

The **-z** *interlv* option lets you specify the amount of interleave (spacing between sectors). Specifying **-z1** (the default) would place the sectors contiguously (e.g. 1,2,3,4,5, etc.)

© 2005, QNX Software Systems

Specifying **-z2** would place the sectors at every second location (e.g. 1, -, 2, -, 3, -, 4, -, 5, etc.). If there were nine sectors per track, **-z2** would yield:

```
1, 6, 2, 7, 3, 8, 4, 9, 5
```

such that reading every other sector in the (circular) track would produce a contiguous reading of the sectors. Note that the hardware makes all of this transparent; specifying **-z** may optimize access for the hardware.

The **-k** *skew_factor* option affects performance when reading and writing consecutive tracks on the disk in an ascending order. Disk drives have an inherent latency when moving the head from one track to the next. By setting the first sector of the next track some distance ahead of the last sector of the current track, it is possible to tune the amount of time it takes before the head encounters the first sector of the next track after the seek. When there is no skew, the rotation of the disk will place the head past the first sector of the next track after the time taken to seek to the next track, meaning that to get to the first sector the disk must complete nearly a full rotation.

If the latency is, for example, five sectors, you would specify the following:

```
-k 5
```

Placing the first sector of the next track five sectors away from the current sector minimizes the latency effects.

Examples:

Format the diskette mounted as **/dev/fd0** using the current mount parameters:

```
fdformat /dev/fd0
```

Format a 1.4M diskette mounted as **/dev/fd0**:

```
fdformat -s 1.4M /dev/fd0
```

Format a 1.4M diskette mounted as **/dev/fd0** using an interleave of 3 because the diskette will be used as a boot diskette:

```
fdformat -s 1.4M -z 3 /dev/fd0
```

Files:

If the **-p** is specified, **fdformat** will pause until gets a newline character from the standard input before proceeding to format the specified disk. The standard input is otherwise unused.

If not in quiet mode (i.e. **-q** is not specified), informational and progress messages will be written to the standard output as the formatting proceeds. The extent of this information is influenced by the **-v** option. If **-q** is specified, this information will not be written. Note that **-q** does not override the printing of the prompt for the **-p** (pause) option.

Any errors which occur will cause a diagnostic message to be written to the standard error. In addition, if the **-v** option is specified, information relating to some of the specific formatting steps may be written to the standard error.

The **fdformat** utility acts upon the block special file named on the command line. The contents of the medium represented by this file will be destroyed on successful invocation of the utility.

Exit status:

0 The disk was formatted successfully.

>0 An error occurred and a diagnostic message was written to the standard error.

Caveats:

The **fdformat** utility will destroy any existing data on a diskette. You should not count on this behavior for data security — differences

in physical drive characteristics and presence of magnetic fields in fringe areas may result in the data being recoverable by use of special instruments. If you want to destroy data, the only really safe way is to completely destroy the media.

Note that the latency saved by tuning the disk skew factor is only realized when reading sequentially across a track boundary. On random requests, there is no single mechanism to minimize access time.

See also:

dcheck, **dinit**, **Fsys**, **Fsys.floppy**, **lockfd**

fdisk

Create QNX disk partition (QNX)

Syntax:

fdisk [-fpz] [-B *loader***]** *drive* **[***cmd***]**

Options:

-B *loader*	Use the 512-byte file named by *loader* as the primary bootstrap loader for the device when instructed to write a boot loader to the disk. The default is to install a loader which is built-in to the **fdisk** utility.
-f	Force boot loader to be written on command, even if it is not possible to save an existing old loader to a mounted filesystem. In non-interactive use with the **loader** command, this will force the loader to be written in cases where the command would otherwise be aborted. In interactive use, the user will be queried about the operation if **-f** is not specified, and the operation will simply proceed without saving the old loader if **-f** is specified.
-p	Pause before starting.
-z	Set the partition table to zero (all entries cleared).
drive	Disk drive to partition.
cmd	An optional installation command, which is one of:

Commands:

add [**-f** *slot*] **{QNX|***type***}**
{ALL|HALF|QUARTER|TENTH|*size***}**
boot [**QNX|QNX2|DOS|OS2|***type***|***slot***]**
info
loader
show
query
[**QNX|QNX2|DOS|OS2|FREE|TOTAL|***type***|***slot***]**

Where:

© 2005, QNX Software Systems

type is the partition specifier (e.g. **t77**), as in the following list:

type	For
t77, **t78**, or **t79**	QNX 4
t7, **t8**, or **t9**	QNX 2.1
t1, **t4**, **t5**, or **t6**	DOS

size is the size of the partition, in cylinders (e.g. **680**) or megabytes (e.g. **72m**).

slot is partition number **1**, **2**, **3**, or **4**.

Refer to the description for more details on partition specifiers.

Description:

The **fdisk** utility lets you partition a hard disk. The partition information matches that used by DOS. The information is kept on the first physical block on the disk.

The **fdisk** utility can be run only by **root** or users with read/write permissions for the block special file concerned.

If you want your disk to contain both QNX and DOS partitions, we recommend that you create the DOS partition *first*, using DOS commands.

WARNING:

You can boot only from OS images that are loaded from within the first 1024 cylinders of the disk. This means that while you may be able to initially install and boot from a partition which extends past the 1024th cylinder, it will someday fail when you go to update the boot image because the location of some of its blocks may change. When this happens you will have a system which is no longer bootable.

Avoid this problem by creating a separate partition to boot from that lies entirely within the first 1024 cylinders of the hard drive, and use a second partition to access the additional space on the drive. (The boot partition may be quite small — just a few megabytes will suffice.)

To create a QNX partition for the first time, you must first start the Filesystem Manager and the hard disk driver. For example:

```
Fsys &
disktrap start
```

You should then execute the **fdisk** command:

```
fdisk /dev/hd0
```

to partition your disk.

QNX doesn't automatically mount any partition. Once you've created the partition, you should mount the disk using this partition (see **mount**).

It's important to realize that **fdisk** only displays and updates the partition information on the disk. It doesn't directly affect your access to the drive (see **mount**).

The **fdisk** utility is a fullscreen, interactive program that's fairly self-explanatory. When you invoke **fdisk**, you'll see a screen similar to this one (assuming your disk is already partitioned):

© 2005, QNX Software Systems

```
FDISK
Ignore Next Prev 1 2 3 4 Change Delete Boot Unboot Restore Loader Save Quit

          __OS__        Start    End     _______Number_____    Size    Boot
        name  type  Cylinder  Cylinder  Cylinders  Blocks
   1.  DOS    (  4)       0       159       160      39165      19 MB
   2.  QNX2.x (  7)     160       419       260      63700      31 MB
   3.  QNX    ( 77)     420       854       435     106575      52 MB   *
   4.  ------ (---)   -----     -----     -----    -------   -----
Choose one of the partitions, by:    typing the partition number OR
moving the pointer with the UP/DOWN arrows; then, choose one of the
actions on the top line of the screen.

                                    Drive :  /dev/hd0

Size  :   102 Mbytes                Config:       7 Heads
Loader:       QNX                                35 Sectors/cyl
                                                855 Cylinders

                                    Last cylinder is 854
```

You'll see the available commands displayed at the top of the screen. To select a command, either type its first letter or move the cursor to the command (with the arrow keys) and press Enter.

The commands are:

Command:	Action:
Next	Move the pointer to the next entry.
Prev	Move the pointer to the previous entry.
1, **2**, **3**, or **4**	Move the pointer to the indicated entry.
Change	Change the selected partition.
Delete	Delete the selected partition.
Boot	Turn on the boot flag for the selected partition.
Unboot	Turn off the boot flag for the selected partition.
Restore	Restore the previous non-QNX bootstrap loader.
Loader	Change the bootstrap loader to the QNX loader.
Save	Save all changes and quit.

continued...

Command:	Action:
Quit	Quit without saving changes.

If you're changing a partition entry, you'll be required to enter the partition type number and the start and end cylinders. The other information will be calculated for you by **fdisk**.

Note that partition types 78 and 79 (which will appear on the interactive display as partition names **QNY**, **QNZ**, respectively) are the suggested OS types to use for second and third QNX 4 partitions.

Command mode:

The **fdisk** utility can also be run as a non-interactive utility using command-line options. This is useful for designing a custom installation procedure.

The command-mode syntax is:

fdisk *drive cmd*

fdisk supports the following commands directly from the command-line:

add [**-f** *slot*] **{QNX|***type***} {ALL|HALF|QUARTER|TENTH|***size***}**

Add a new partition entry of the size and type specified. If **fdisk** cannot locate sufficient unallocated disk space to satisfy your request, it will allocate the largest available portion of the disk (if any).

Using the **-f** (force) option deletes any existing partition in the specified partition number of the partition table. All allocated blocks are freed for reuse before attempting to allocate the requested space. Also, **add** will normally use the first available partition number, but the **-f** option will force it to use *slot*, the specified number.

© 2005, QNX Software Systems

boot [**QNX**|**QNX2**|**DOS**|**OS2**|*type*|*slot*]
: Turn on the boot flag for the indicated partition. If another partition was already flagged as the primary boot partition, the flag will be turned off for it.

info
: Show the mount information for the raw drive.

loader
: Write the QNX loader to the disk.

show
: Show the **fdisk** partition entries.

query [**QNX**|**QNX2**|**DOS**|**OS2**|**FREE**|**TOTAL**|*type*|*slot*]
: Print the number of cylinders to standard output. Where:

type
: is the partition specifier (e.g. **t77**), as in the following list:

t*type*	For
t77, **t78**, or **t79**	QNX 4
t7, **t8**, or **t9**	QNX 2.1
t1, **t4**, **t5**, or **t6**	DOS

size
: is the size of the partition, in cylinders (e.g. **680**) or megabytes (e.g. **72m**).

slot
: is partition number **1**, **2**, **3**, or **4**.

Examples:

Interactively create or edit the partition table of **/dev/hd0**:

```
fdisk /dev/hd0
```

Create a QNX partition which occupies half the disk, or the largest available space if there isn't a space big enough for a new partition that occupies half the disk:

```
fdisk /dev/hd0 add qnx HALF
```

Files:

The *drive* specified on the command line must name a block special file.

When no *cmd* is specified on the command line, **fdisk** is an intensely interactive fullscreen program using the standard input and standard output. The standard error is used for diagnostic messages upon severe errors (in non-interactive mode) and for ASCII BEL characters in interactive mode used to notify the user that something invalid has been entered into a field.

Exit status:

0 Successful.

>0 An error occurred.

Caveats:

After changing *any* partition that is **mount**ed, you must either **umount** and then re-**mount** the changed partition or simply reboot. If you don't take these steps, you could damage the filesystem on the new partition.

See also:

dinit, **fdformat**, **Fsys**, **mount**

fgrep

© 2005, QNX Software Systems

Fixed string grep (UNIX)

Syntax:

fgrep [-chilnqsvx] [-e *expression*|**-f** *expression_file*]... **[***file***...]**

fgrep [-chilnqsvx] *expression* **[***file***...]**

Options:

See **grep**.

Description:

The **fgrep** utility does a fixed-string **grep** (equivalent to **grep -F**). See **grep** for details.

Exit status:

0 Lines were found matching the expression provided.

>0 An error occurred or no matching lines were found.

See also:

grep, **diff**, **find**, **sort**

Syntax:

file [-vczL] [-f *namefile***] [-m** *magicfile***]** *file* **...**

Options:

-c	Cause a checking printout of the parsed form of the **magic** file. This is usually used in conjunction with **-m** to debug a new magic file before installing it.
-f *namefile*	Read the names of the files to be examined from *namefile* (one per line) before the argument list. Either *namefile* or at least one filename argument must be present; to test the standard input, use **-** as a filename argument.
-L	Cause symlinks to be followed, as the like-named option in **ls**.
-m *file*	Specify an alternate file of magic numbers.
-v	Print the version of the program and exit.
-z	Try to look inside compressed files.

Description:

The **file** utility tests each *file* argument in an attempt to classify it. There are three sets of tests, performed in this order:

1 Filesystem tests

2 Magic number tests

3 Language tests

The *first* test that succeeds causes the file type to be printed. The type printed usually contains one of these words:

text	The file contains only ASCII characters and is probably safe to read on an ASCII terminal.

© 2005, QNX Software Systems

executable The file contains the result of compiling a program in a form understandable to some UNIX kernel or another.

data Meaning anything else (data is usually "binary" or nonprintable). Exceptions are well-known file formats (**core** files, **tar** archives) that are known to contain binary data.

When modifying the file **/etc/magic** or the program itself, *preserve these keywords*. People depend on knowing that all the readable files in a directory have the word **text** printed.

The filesystem tests are based on examining the return from a *stat()* system call. The program checks to see if the file is empty, or if it's some sort of special file. Any known file types appropriate to the system you're running on (sockets, symbolic links, or named pipes (FIFOs) on those systems that implement them) are intuited if they're defined in the system header file **/usr/include/sys/stat.h**.

The magic number tests are used to check for files with data in particular fixed formats. These files have a "magic number" stored in a particular place near the beginning of the file that tells the UNIX operating system that the file is a binary executable, and which of several types thereof. The concept of "magic number" has been applied by extension to data files. Any file with some invariant identifier at a small fixed offset into the file can usually be described in this way. The information in these files is read from the magic file **/etc/magic**.

If an argument appears to be an ASCII file, **file** attempts to guess its language. The language tests look for particular strings that can appear anywhere in the first few blocks of a file. For example, the keyword **.br** indicates that the file is most likely a **troff** input file, just as the keyword **struct** indicates a C program. These tests are less reliable than the previous two groups, so they are performed last. The language test routines also test for some miscellany (such as **tar**

archives) and determine whether an unknown file should be labeled as **ascii text** or **data**.

Files:

/etc/magic Default list of magic numbers.

Author:

Written by Ian F. Darwin, UUCP address **{utzoo|ihnp4}!darwin!ian**, Internet address **ian@sq.com**, postal address: P.O. Box 603, Station F, Toronto, Ontario, CANADA M4Y 2L8.

Altered by Rob McMahon, **cudcv@warwick.ac.uk**, 1989, to extend the **&** operator from simple **x&y != 0** to **x&y op z**.

Altered by Guy Harris, **guy@auspex.com**, 1993, to:

- Put the "old-style" **&** operator back the way it was, because 1) Rob McMahon's change broke the previous style of usage, 2) the SunOS "new-style" **&** operator, which this version of **file** supports, also handles **x&y op z**, and 3) Rob's change wasn't documented in any case;
- Put in multiple levels of **>**
- Put in **beshort**, **leshort**, etc. keywords to look at numbers in the file in a specific byte order, rather than in the native byte order of the process running **file**.

Changes by Ian Darwin and various authors including Christos Zoulas (christos@ee.cornell.edu), 1990-1992.

License:

Copyright © Ian F. Darwin, Toronto, Canada, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993.

This software is not subject to and may not be made subject to any license of the American Telephone and Telegraph Company, Sun

Microsystems Inc., Digital Equipment Inc., Lotus Development Inc., the Regents of the University of California, The X Consortium or MIT, or The Free Software Foundation.

This software is not subject to any export provision of the United States Department of Commerce, and may be exported to any country or planet.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

A few support files (*getopt(), strtok()*) distributed with this package are by Henry Spencer and are subject to the same terms as above. A few simple support files (*strtol(), strchr()*) distributed with this package are in the public domain; they are so marked.

The files **tar.h** and **is_tar.c** were written by John Gilmore from his public-domain **tar** program, and are not covered by the above restrictions.

Source:

You can obtain the original author's latest version by anonymous FTP on **tesla.ee.cornell.edu** in the directory **/pub/file-X.YY.tar.gz**.

Caveats:

The **file** utility uses several algorithms that favor speed over accuracy, thus it can be misled about the contents of ASCII files. The support for ASCII files (primarily for programming languages) is simplistic, inefficient and requires recompilation to update.

See also:

/etc/magic

find

© 2005, QNX Software Systems

Find files (POSIX)

Syntax:

find *path...* **[***operand_expression***]**

find [*limited_operand_expression***]**

Options:

path... This names the pathnames under which **find** should search for files. The utility will traverse the pathname tree from these files down, looking for files which match the search criteria specified by the *operand_expression*. If no *path*s are specified, and the expression meets the criteria of a *limited_operand_expression* (below), then a *path* of **.** is assumed.

operand_expression

This is an expression composed of any set of the primary expressions and operators described in the description.

limited_operand_expression

This represents an expression composed of any of the primary expressions and operators described in the description *except* for **-exec**, **-ok** and **-spawn**.

If no *operand_expression* or *limited_operand_expression* is supplied on the command line, an *operand_expression* of **-print** is assumed.

Description:

The **find** utility recursively descends the directory hierarchy for each file specified by *path* and seeks files that match *operand_expression*.

When an operand expression isn't specified, the **find** utility will match every file and directory, printing each pathname to the standard output, one file per line.

The operand expression follows one or more pathnames. The **find** utility treats as a pathname all the arguments up to the first one starting with any of these characters:

-, **!**, **(**

Everything after that is part of the operand expression.

Operand expressions are made up of primaries and operators. The following operators are supported:

Operator	Action
!	(**NOT**)
-a	(**AND**)
-o	(**OR**)

AND operations have higher precedence than **OR** operators. Negation (**!**) has a higher precedence than **AND** operators. Parentheses are supported to override the normal precedence rules.

The rules that apply to primaries and operators can be summarized as follows:

Expression	Evaluates to
-*primary*	True if *primary* is true.
(*expression* **)**	True if *expression* is true.
! *expression*	(**NOT**) Negation of a primary or expression enclosed in parentheses.

continued...

© 2005, QNX Software Systems

Expression	Evaluates to
expression [**-a**] *expression*	(**AND**) True if both expressions are true. If the first expression is false, the second expression won't be evaluated. The **-a** is optional. **AND** is implied by the juxtaposition of two expressions.
expression **-o** *expression*	(**OR**) True if either expression is true. If the first expression is true, the second expression won't be evaluated.

Primary expressions

Note that when no expression is supplied, **find** behaves as if an expression of **-print** were specified.

If an expression is supplied but it doesn't contain a **-chmod**, **-chown**, **-exec**, **-fls**, **-fprint**, **-fprint0**, **-fprintf**, **-ls**, **-ok**, **-print**, **-print0**, **-printf**, **-rename**, **-remove!**, or **-spawn** primary, the **find** utility operates as if the following expression were supplied:

(*given_expression*) -print

Specifying "more than" or "less than" with numerical arguments

Whenever a number *n* is used in a primary expression, it may be optionally preceded by a plus (**+**) or minus (**-**) sign, which changes the meaning as follows:

Expression	Means
+*n*	more than *n*
-*n*	less than *n*

continued...

Expression	Means
n	exactly *n*

Each of the primaries listed below have been identified as **POSIX**, **QNX** and/or **GNU** for the benefit of those concerned with portable use of the **find** utility.

POSIX the primary is supported by any POSIX **find** implementation

GNU the primary is supported by the QNX and GNU **find** implementations

QNX the primary is supported only by the QNX implementation

-abort (QNX) Immediately terminate the find with a nonzero exit status.

-amin *n* (GNU) True if the file was last accessed *n* minutes ago.

-anewer *file* (GNU) True if the file being evaluated was accessed more recently than *file*.

-atime *n* (POSIX) True if the file access time subtracted from the time that the **find** utility started running is between *n*-1 and *n* multiples of 24 hours. For example, **find . -atime 1** will find all files under the current directory for which the file was accessed within the last 24 hours.

-chgrp *gname* (QNX) Changes the file group ownership of the file currently being evaluated to *gname*. If *gname* is numeric and the name doesn't exist in the group database, *gname* is taken as a group ID.

Specifying **-chgrp** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-chmod *mode* (QNX) Changes the permissions of the file currently being evaluated according to the specified *mode*.

The *mode* argument represents file mode bits. It is identical to the *symbolic_mode* operand described in **chmod**.

For example, to remove write permission for group and other from the file being evaluated, you would use **-chmod go-w**.

Specifying **-chmod** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-chown *uname*[:*gname*]

(QNX) Changes the file ownership of the file currently being evaluated to the one specified, taking a ownership specification similar to that accepted by the **chown** utility. A user name *uname* must be specified, optionally followed by a colon (**:**) and group name *gname*. The *uname* and *gname* parameters may be either an ASCII name or the actual numeric user or group ID.

Specifying **-chown** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-cmin *n* (GNU) True if the file status was last changed *n* minutes ago.

-cnewer *file* (GNU) True if the file being evaluated had its status changed more recently than that of *file*.

-ctime *n* (POSIX) True if the file change of status time subtracted from the time that the **find** utility started running is between *n*-1 and *n* multiples of 24 hours.

-daystart (GNU) Always true. When used, this primitive will cause **find** to globally alter the behavior of the **-atime**, **-ctime**, **-ftime**, and **-mtime** primitives: instead of comparing file times to *n* 24-hour periods before the current time, it will compare file times to *n* 24-hour periods before the beginning of the current calendar day.

-depth (POSIX) Always true; causes descent of the directory hierarchy to be done so that all entries in the directory are acted on before the directory itself. If no **-depth** primary is specified, then all entries in the directory are acted on after the directory itself. If any **-depth** primary is specified, then it applies to the entire expression even if the **-depth** primary wouldn't normally be evaluated.

-echo [*text*] **;** (QNX) Write the supplied text to standard output. If the text contains any **{}**s, they're interpreted as in the **-exec** primitive to represent the pathname being evaluated.

Specifying **-echo** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-empty (GNU) True if the file is a regular file of size 0, or a directory which contains no files.

-errmsg [*text*] **;** (QNX) Similar to **-echo**, except the output is written to standard error.

Specifying **-errmsg** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-error (QNX) Cause the exit status to be nonzero when the find is completed. This primary always evaluates to false and is typically used with **-exec**. For example, if you enter this command:

```
find /bin -type f \( -exec cmp {} //9/hd{} \; \
     -o -error \)
```

find will exit with a nonzero status if any of the files in **/bin** doesn't compare successfully against the same files under **//9/hd/bin**.

-exec *utility_name* [*argument...*] **;**

(POSIX) True if the executed utility *utility_name* returns a zero value as exit status. The end of the primary expression is punctuated by a semicolon (**;**).

If a *utility_name* or *argument* contains **{**...**}**, the **{**...**}** is replaced by the current pathname or a portion thereof as follows:

A **{}** in a *utility_name* or *argument* will be replaced by the pathname being evaluated. Such arguments are used by **-echo**, **-errmsg**, **-exists**, **-fanewer**, **-fcnewer**, **-fFnewer**, **-fmnewer**, **-fnewer**, **ok**, **-rename** and **-spawn** in addition to **-exec**.

The current directory for the execution of *utility_name* is the same as the current directory when the **find** utility was started.

Specifying **-exec** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

There is a QNX-only extension to the **{}** syntax for stripping leading and trailing characters. To insert the filename stripped of a number of characters at the end (*strip*) or the filename less a number of characters at the beginning (*skip*), te syntax is

{**[***strip***][,***skip***]**}

So, to move all files ending in **.c** to the same names ending in **.C**, one might use:

```
find . -type f -name '*.c' -exec 'mv {} {1}C' \;
```

-exists *filestring*

(QNX) True if the file represented by *filestring* exists. *filestring* may be a simple filename or it may contain **{}**s to represent the name of the file currently being evaluated, in the same manner as used with the **-exec** primitive.

-extents *n* (QNX) True if the file has *n* extents in the filesystem.

-false (GNU) Always false.

-fanewer *filestring*

(QNX) True if the file was accessed more recently than the file represented by *filestring*.

-fcnewer *filestring*

(QNX) True if the file had its status changed more recently than the file represented by *filestring*.

-fFnewer *filestring*

(QNX) True if the file was created more recently than the file represented by *filestring*.

-fls *file* (GNU) Always true. Similar to **-ls**, but output is written to the file *file* instead of the standard output.

Specifying **-fls** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-fmnewer *filestring*

(QNX) True if the file was modified more recently than the file represented by *filestring*.

-fnewer *filestring*

(QNX) Synonym for **-fmnewer** (above).

-Fnewer *file* (QNX) True if the file being evaluated was created more recently than *file*.

-follow (GNU) Always true. When **-follow** is specified, **find** will treat symbolic links as being the type of the file they point to. If the link points to a directory, **find** will recurse into that directory. By default, symbolic links are treated as special files of type symbolic link. What they point to is irrelevant to **find**'s default behavior.)

-fprint *file* (GNU) Similar to **-print**, but output is written to *file* instead of to the standard output.

Specifying **-fprint** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-fprint0 *file* (GNU) Similar to **-print**, but each file name is followed by a NUL instead of a newline character, and the output is written to *file* instead of to the standard output.

Specifying **-fprint0** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-fprintf *file format*

(GNU) Always true. Write data pertaining to the file currently being evaluated to *file*, according to the *format* specified. See "Formatted Printing" for details on the format string.

Specifying **-fprintf** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-fsmanager *name*

(QNX) True if the file being evaluated is being managed by a process whose name, as reported by its response to a version query, matches *name*. Typical names include **Fsys32**, **Fsys**, **Dev32**, **Dev16**, **Proc**, **Pipe**, **Iso9660fsys**, **Photon**, **X11R5**, **Mouse**, **Socket**, and **INT10h**. A special name "**Unknown**", will match files whose managers reject the version query.

-ftime *n* (QNX) True if the file creation time subtracted from the time that the **find** utility started running is between *n*-1 and *n* multiples of 24 hours.

-gid *n*|*groupname*

(GNU) Synonym for **-group**.

-group *gname* (POSIX) True if the file belongs to the group *gname*. If *gname* is numeric and the name doesn't exist in the group database, *gname* is taken as a group ID. Note that *gname* is evaluated only once.

-ilname *fpattern*

(GNU) Similar to **-lname**, but case-insensitive in the pattern match.

-iname *fpattern* (GNU) Similar to **-name**, but case-insensitive in the pattern match.

-inode *file*|*n* (QNX) True if the file uses the same inode (has the same serial number) as the named *file*. If the file doesn't exist and is numeric, the number is used as the serial number to match. This primary is used to find links to a file.

-inum *n*|*file* (GNU) Synonym for **-inode**.

-ipath *fpattern* (GNU) Like **-path**, but case-insensitive when evaluating the pattern match.

-iregex *ere* (GNU) Like **-regex**, but case-insensitive when evaluating the regular expression match.

-level *n* (QNX) True when the level down in a directory tree is *n*. A file or directory specified on the command line is considered level 0. For example, this command:

```
find /usr -level 1 -type d -print -o \
          -level 2 -prune -type f \
          -name .usrinit -ls
```

will display all the directories in **/usr**, and for each directory that has a **.usrinit** file, will display information on that file in **ls -l** format. (The **-prune** at level 2 prevents unnecessary processing in walking down the directory tree. Though no files further down could possibly match the **-level 1** or **-level 2** criteria, **find** doesn't detect this automatically — the command-line expression is applied against every file in the directory tree unless a full recursion of that tree is prevented by a **-prune** primitive.)

The following command:

```
find /usr -level 1 -ls -prune
```

will display information in **ls -l** format only on files in **/usr** and won't descend into any subdirectories of **/usr**.

-links *n* (POSIX) True if the file has *n* links.

-lname *fpattern* (GNU) True if **-follow**/**-logical** is not specified, and the file being evaluated is a symbolic link whose target is a pathname which matches the pattern *fpattern*.

-logical (QNX) Synonym for **-follow**.

-ls (GNU) Similar to **-print**, but displays in the same format as **ls -l**.

Specifying **-ls** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-maxdepth *n* (GNU) Always true. When this flag is set, **find** will descend at most *n* levels in the directory hierarchy. Files that are named on the command line are level 0. Note: the **+** and **-** modifiers have no meaning when used in conjunction with *n* in this primary.

-mindepth *n* (GNU) Always true. When this flag is set, **find** will not apply the expression to files unless the files are at least *n* levels down in the directory hierarchy. Files named on the command line are level 0. Note: the **+** and **-** modifiers have no meaning when used in conjunction with *n* in this primary.

-mmin *n* (GNU) True if the file data was last modified *n* minutes ago.

-mnewer *file* (QNX) True if the file being evaluated was modified more recently than *file* was.

-mount (GNU) Synonym for **-xdev**.

-mountdev (QNX) True if the file is a block special file for which there is a corresponding filesystem mount point.

-mountpoint (QNX) True if the file is a directory that is the mount point for a QNX filesystem. If you just want to find all mount points, it's faster to use **df** or **prefix** instead.

-mtime *n*
(POSIX) True if the file modification time subtracted from the time that the **find** utility started running is between *n*-1 and *n* multiples of 24 hours.

-name *pattern*
(POSIX) True if the basename of the filename being examined matches *pattern*. This follows the same pattern matching rules as used by the *fnmatch()* C function. (For details, see the section on "Pattern-matching special characters" in the **cp** utility.)

-newer *file*
(POSIX) True if the current file has been modified more recently than the modification time of the file named by the *file* pathname. Note that *file* is evaluated only once.

-nogroup
(POSIX) True if the file belongs to a group ID that isn't in the group database.

-NOP
(QNX) Always true, does nothing. This primitive has the side effect of disabling the implicit **-print** that occurs when the expression as a whole evaluates to true. This primitive can be used to benchmark the time it takes to do a walk of the filesystem. For example:

```
find / -NOP
```

-nouser
(POSIX) True if the file belongs to a user ID that isn't in the password database.

-ok *utility_name* [*argument*...] **;**
(POSIX) Similar to **-exec**, except that **find** requests affirmation of the execution of *utility_name* using the current file as an argument by writing to standard error. If the response on standard input is affirmative, the utility is

executed. If the response isn't affirmative, the command isn't executed and the value of the **-ok** operand is false.

Specifying **-ok** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-path *fpattern* (GNU) True for any file whose path (as would be printed by **-print**) matches *fpattern*.

-perm [**-**]*mode* (POSIX) The *mode* argument represents file mode bits. It is identical to the *symbolic_mode* operand described in **chmod** and is interpreted as follows. To start, a template is assumed with all file mode bits cleared. An *op* symbol of:

Operator:	Action:
+	sets the appropriate mode bits
-	clears the appropriate mode bits
=	sets the appropriate mode bits, regardless of the process's file mode creation mask

An *op* symbol of **-** cannot be the first character of *mode*.

If the optional hyphen preceding *mode* is omitted, the primary is true when the file permission bits exactly match the value of the resulting template. In addition, the bits associated with the *perm* symbol **s** are ignored. If the hyphen is included, the primary is true if at least all the bits in the resulting template are set.

-perm [**-**] *onum* (POSIX) If the optional hyphen is omitted, the primary is true when the file permission bits exactly match the value of the octal number *onum*

and only the bits corresponding to the octal mask 777 are compared.

If the hyphen is included, more flag bits, corresponding to the octal mask 06777, are compared and the primary is true if at least all the bits in *onum* are set.

-pname *pattern* (QNX) Synonym for **-path**.

-print (POSIX) Always true; causes the current pathname to be written to standard output, one pathname per line.

Specifying **-print** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-print0 (GNU) Always true. Writes the path currently being evaluated followed by an ASCII NUL character to the standard output.

Specifying **-print0** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-printf *format* (GNU) Always true. Write data pertaining to the file currently being evaluated to the standard output, according to the *format* specified. See the "Description" section for details on the format string.

Specifying **-printf** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-prune (POSIX) Stops **find**'s descent from that point in the file hierarchy.

-regex *ere* (GNU) True when the pathname of the file currently being evaluated matches the extended regular expression specified by *ere*. See the **grep**

utility documentation for details on regular expressions.

-remove! (QNX) Removes the file being evaluated. If the file is a directory, *rmdir()* is performed, otherwise an attempt is made to *unlink()* the file. If a directory is not empty the attempt to remove it will fail. Thus, to recursively remove a directory tree with **find**, the **-depth** primitive must be used in conjunction with **-remove!**. (Note the simple removal of a directory tree is better and more portably done by using the **rm** utility.) This primitive evaluates to TRUE if the removal was successful. Note that the exclamation mark (**!**) is a required part of this primitive.

Specifying **-remove!** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-rename *filestring*

(QNX) Renames the file to the pathname indicated by *filestring*. As with other *filestring* arguments, **{}**s encountered in the string will be expanded to the name of the file currently being evaluated. A file may be renamed anywhere within the same filesystem. If the new path lies on another device the rename will fail. Evaluates to true if the rename succeeds, false if it fails.

Specifying **-rename** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-size *n*[**c**] (POSIX) True if the file size in bytes, divided by 512 and rounded up to the next integer, is *n*. If *n* is followed by **c**, the size is in bytes.

-spawn *cmd* [*arguments*]... **;**

(QNX) Similar to **-exec**, except that the command is invoked directly (i.e. not through a shell). The **-spawn** primary is faster but less flexible than **-exec**.

Specifying **-spawn** will inhibit the automatic **-print** when the expression as a whole evaluates to true.

-status [**grown**|**busy**]

(QNX) True if the file has a pre-grown or busy status, as specified.

-true (GNU) Always true.

-type *c* (POSIX) True if the filetype is *c*, where *c* can be:

Filetype:	Description:
b	block special file
c	character special file
d	directory
p	FIFO
f	regular file
l	symbolic link
n	named special file

-uid *n*|*userid* (GNU) Synonym for **-user**.

-used *n* (GNU) True if file was last accessed *n* days after its status was last changed.

-user *uname* (POSIX) True if the file belongs to the user *uname*. If *uname* is numeric and the name doesn't

exist in the password database, *uname* is taken as a user ID. Note that *uname* is evaluated only once.

-xdev (POSIX) Always true — stops **find** from descending past directories that have a different device ID. If any **-xdev** primary is specified, it applies to the entire expression even if the **-xdev** primary wouldn't normally be evaluated.

Formatted printing (-printf and -fprintf primitives)

The **-printf** and **-fprintf** primaries require as arguments a format string similar in appearance to that used in the C language *printf()* function. The format string consists of regular ASCII characters and a set of special codes starting with percent (%) format codes and backslash (\) escape codes.

printf Backslash (\) Escape Codes

**** Literal backslash (\) character.

\a Alarm bell.

\b Backspace character.

\c Stop printing from format and flush output.

\f Form-feed character.

\n Newline character.

\r Carriage return character.

\v Vertical tab character.

printf Format Codes

%% Literal percent (%) character.

%a*fchar* Partial or full representation of the file's last access time, depending on the format character *fchar*

© 2005, QNX Software Systems

fchar:	Is replaced by:
a	abbreviated weekday name
A	full weekday name
b	abbreviated month name
B	full month name
c	locale's appropriate date and time representation
d	day of the month as a decimal number (01-31)
D	date in the format mm/dd/yy
h	abbreviated month name
H	hour (24 hr) as a decimal number (00-23)
I	hour (12 hr) as a decimal number (01-12)
j	day of the year as a decimal number (001-366)
m	month as a decimal number (01-12)
M	minute as a decimal number
n	newline character
p	AM or PM
r	12-hr clock time (01-12) using the AM/PM notation i.e. hh:mm:ss (AM|PM)
S	second as a decimal number (00-59)
t	tab character
T	24-hr clock time (00-23) hh:mm:ss
U	week number of the year as a decimal number (00-52) where Sunday is the first day of the week
w	weekday as a decimal number (0-6) where 0 is Sunday

continued...

fchar: Is replaced by:	
W	week number of the year as a decimal number (00-52) where Monday is the first day of the week
x	locale's appropriate date representation
X	locale's appropriate time representation
y	year without century as a decimal number
Y	year with century as a decimal number
Z	timezone name, NULL if no timezone exists

%b The file's size in 512-byte blocks (rounded up).

%c file's last status change date and time — equivalent to **%Cc**

%C*fchar* Partial or full representation of the file's last status change time, depending on the format character *fchar*. See description of **%a**, above for details.

%d The depth of the file in the directory tree. The files specified on the command line have a depth of 0.

%D The name of the mounted block-special device that the file resides in.

%f The basename of the file.

%F The name of the manager process responsible for the file.

%g The file's group name, or numeric group ID if no name found.

%G The file's numeric group ID.

%h The dirname of the file.

%H The name of the root of the file tree (pathname specified on the command line under which the current file was found).

%i	The inode number of the file.
%k	The file's size in 1K units (rounded up).
%l	If the file is a symbolic link, the filename of the link object. Nil if file is not a symbolic link.
%m	The file's permissions in octal.
%M	The mount point of the filesystem which the file resides on.
%n	The link count of the file.
%p	The pathname currently being evaluated.
%P	The name of the file with the root of the file tree (pathname specified on the command line) removed from the beginning.
%s	The file's size in bytes.
%t	file's last modification date and time — equivalent to **%Tc**
%T*fchar*	Partial or full representation of the file's last modification time, depending on the format character *fchar*. See description of **%a**, above for details.
%u	The file's user name, or numeric user ID if no name found.
%U	The file's numeric user ID.

Examples:

Search the filesystem for the **myfile** file or directory:

```
find / -name myfile
```

Remove all files named **tmp** or ending in **.xx** that haven't been accessed for seven or more 24-hour periods:

```
find / \( -name tmp -o -name '*.xx' \) \
     -atime +7 -exec rm {} \;
```

Print the pathnames of all files in and below the current directory, but skip any directories named **SCCS** and the files beneath them:

```
find . -name SCCS -prune -o -print
```

Note that when possible, it's better to use **find** in combination with **xargs** than it is to use the **-exec** or **-spawn** options to start commands. The **xargs** utility can be used to start a program once for many files; **-exec** or **-spawn** will start the program once for every file matched. You'll see a tremendous difference in speed between the two approaches. For instance:

```
find / -name '*.tmp' | xargs rm
```

is generally preferable to:

```
find / -name '*.tmp' -exec rm {} \;
```

See **xargs** for more detail.

Exit status:

0 At least one file was found which matched the search criteria.

>0 An error occurred, or no files were found matching the search criteria.

Caveats:

If **-exec**, **-ok**, or **-spawn** is used, **find** will catch SIGINT (which can be generated, for example, by Ctrl – C or Ctrl – Break) and will ask you whether the find should continue. If these primaries aren't used, **find** will terminate from SIGINT.

© 2005, QNX Software Systems

See also:

chmod, **chown**, **mv**, **pax**, **rm**, **xargs**

finger

Display information about a user (UNIX)

Syntax:

finger [-lmps] [*user* **|** *user@host***]...**

Options:

-l	("el") (default) Produces a multi-line format displaying all of the information described for the **-s** option as well as the user's home directory, phone number, login shell, mail status, and the contents of the files **.plan** and **.project** from the user's home directory.
-m	Match login user names only, not real names. By default **finger** will match login names and the users' real names, resulting in multiple matches when the login name is a part of other users' real names. All name matching is case insensitive.
-p	Don't print the contents of the **.plan** and **.project** files.
-s	Display the user's login name, real name, terminal name and write status (a ***** after the terminal name indicates that write permission is denied), idle time, login time, office location and office phone number. This information will be written in a single line for each session belonging to the user matched.
user	The login ID of a user.
user@host	The login ID of a user on the named host. This form requires the optional TCP/IP for QNX product to work.

Description:

The **finger** utility displays information about a user. The information displayed can include such things as the the login name (i.e. the supplied *username*), the user's real name, the user's root directory, terminal name and write status, idle time, office location,

office phone number, home phone number, login shell, and mail status. It is common for some of this information to be unavailable for any given user.

By default the **finger** utility also displays the following:

- a "project" — the contents of a file called **.project**
- a "plan" — the contents of a file called **.plan**. These files are displayed only if they exist in the named user's home directory and if the invoking user has read permission.

  The full user name in the *In real life* field is the first portion of the *comment* field for that user in the **/etc/passwd** file, up to the first comma in that field (or up to the end of the field if there's no comma). Additional information after a comma is displayed on a separate comment line.

When no *user* or *user@host* is specified, all users logged in to the system are displayed in **-s** format.

When *@host* is specified, the users currently logged in to *host* will be listed in **-s** format.

Examples:

Display information about user **eric**:

```
finger eric
```

Files:

/etc/passwd

Password file; defines userids in use on the system and home directory locations.

/etc/acclog

Account log; tracks login time information (done by **login**).

(*username*'s home directory)**/.lastlogin**

Information on last login, includes tty.

(*username*'s home directory)**/.project**

User's project description (one line)

(*username*'s home directory)**/.plan**

User's plan (multiple lines)

See also:

passwd, **login**

fold

© 2005, QNX Software Systems

Fold lines (POSIX)

Syntax:

fold [-bs] [-w *width***] [***file***...]**

Options:

-b Count width in bytes rather than column positions. If **-b** is specified, **carriage-return**, **backspace** and **tab** are treated as normal characters. Only **newline** remains special.

-s If a segment of a line contains a blank within the first *width* column positions, break the line after the last such blank meeting the width constraints (i.e. avoid breaking the line in the middle of a word).

-w *width* Specify the maximum line length. The default is 80 characters.

Description:

The **fold** filter folds lines in files by breaking up lines that have a width in excess of 80 or, if specified, the *width* in the command-line **-w** option. Lines are folded by the insertion of a **newline** character. All output is written to the standard output.

In all cases, the current count of line width is reset to zero when a **newline** character is encountered in the input. In addition, when the **-b** option is not specified, the following characters have a special meaning when encountered in the input:

carriage-return

The current count of line width is set to zero.

backspace The current count of line width is decremented by one (but it will not go below zero).

tab The current count of line width is incremented to the next count for which *count* modulo 8 equals 1.

Examples:

Fold the file **myfile.txt** to a maximum line width of 80 characters, making the line breaks on the last word boundary (white space) before the width was exceeded and write the results to the standard output:

```
fold -s myfile.txt
```

Fold the file **myfile.txt** to a width of 40 characters, and treat all characters except **newline** as occupying one character (byte) position and write the results to the standard output:

```
fold -bw40 myfile.txt
```

Files:

If no *file*s are specified on the command line, **fold** reads lines to be folded from the standard input until **EOF** is reached.

The **fold** utility writes all folded output from all input files to the standard output.

If an error occurs, a diagnostic message will be written to the standard error.

The **fold** utility will read lines from the text files specified on the command line. If the **-b** is specified, the input files need not be text files.

Exit status:

>0	An error occurred.
0	All files were processed successfully.

See also:

pr, **sed**

freeze

© 2005, QNX Software Systems

Compress and uncompress files (UNIX)

Syntax:

freeze [-cdfvV] [[+*n1*,*n2*,...,*n8*] [*filename*]]...

Options:

-c Write the results of the freeze/melt operation to standard output. No files are changed.

-d Uncompress the files from the archive (**melt**).

-f Force creation of the **.F** file. This option creates the **.F** file, even if one already existed, without prompting you for confirmation. The **.F** file will be created even if the compressed file is larger than the original.

-v Display a counter. For each file being frozen, display a kilobytes counter, and, after the file is frozen, print a message giving the percentage of the input file's size that has been saved by compression.

-V Display the program's version number and compilation options.

-z If the output of a melt (option **-d**) is to a tty, allocate a larger output buffer so screen output is in fullscreen chunks.

+*n1*,*n2*,...,*n8*

A list of 8 numbers separated by commas, specifying the values for the static Huffman table.

Description:

The **freeze**/**melt**/**fcat** compression utilities will eventually become deprecated in favor of the GNU **zip** suite, **gzip**/**gunzip**/**zcat**. The **freeze** suite of utilities will continue to be provided for quite some time before being eliminated completely.

The **freeze** utility compresses the specified files or the standard input. If a file becomes smaller, it's replaced by a file with the extension **.F** (you can use the **-f** option to force the creation of the **.F** file, even if the compressed file is larger). If no files are specified, the compression is applied to the standard input and is written to the standard output.

If you don't specify the **-f** option, and you run **freeze** in the foreground, you're prompted as to whether the file should be overwritten.

CAUTION:

When a **.F** file is created, the original file is removed.

Normally there will be several links set up to the **freeze** utility which behave differently when invoked under the following command names:

This link:	is equivalent to:
melt	**freeze -d**
fcat	**freeze -cd**

You can restore compressed files to their original form through either of two ways:

- specifying the **-d** option to **freeze**
- running **melt** on the **.F** files or on the standard input

When you specify filenames, **freeze** maintains the ownership, modes, access times, and modification times between the file and its **.F** version. As a result, **freeze** can be used for archival purposes, yet can still be used with **make** after melting.

The **freeze** utility uses the Lempel-Ziv algorithm on the first pass and the dynamic Huffman algorithm on the second pass. The size of the "sliding window" is 8K and the maximum length of "matched string" is 256. The positions on the window are coded using a static Huffman table.

A two-byte magic number is prepended to the file to ensure that neither melting of random text nor refreezing of already frozen text is attempted. In addition, the characteristics of the static Huffman table being used during the compression are written to the file so that these characteristics may be reused for other files via the "**+***n1*,*n2*,...,*n8*" command line argument.

The amount of compression you obtain depends on the size of the input file and the distribution of character substrings and their probabilities. Typically, text files (e.g. C programs) are reduced by 60% to 75%, while executable files are reduced by 50%. Compression is generally much better than that achieved by LZW coding or by Huffman coding, although it takes more time to compute.

An argument preceded by a **+** defines the values to use for the Huffman table. You may want to explicitly state these values in order to modify the compression algorithm.

Examples:

Freeze all files in the current directory:

```
freeze *
```

Expand files from a compressed pax archive on floppy:

```
vol -r /dev/fd0 | melt | pax -r
```

View the concatenated contents of all compressed files in the current directory:

```
fcat *.F | more
```

Exit status:

0 Normal exit.

1 An error occurred.

2 The last file's compressed size was greater than its original uncompressed size.

Author:

Leonid A. Broukhis

See also:

cpio, **fcat**, **gunzip**, **gzip**, **melt**, **pax**, **tar**, **zcat**

Fsys

© 2005, QNX Software Systems

Filesystem manager (QNX)

Syntax:

Fsys [*options***] &**

Options:

-a Perform all system writes (inodes, bitmap, etc.) asynchronously.

-A Don't update access times if nothing else about the file (mode, owner, data etc.) changed. If this option is used, the filesystem will not behave in the manner required by POSIX. Utilities and scripts which rely on time of last access will not behave as would be expected on a POSIX system. The advantage of **-A** is that it is slightly faster on a heavily loaded server.

-C *num_names* Number of pathname components to cache. The default is 1.5x the number of in-memory inodes (specified by **-i**). If you increase this value, you may also need to increase **-i** accordingly.

-c *cache_size*[**m|k|b|%**]

The size of the cache. Units are assumed to be 1K unless the units are specified as follows:

Char	Unit
m or **M**	megabytes
k or **K**	kilobytes
b or **B**	512-byte blocks
%	Percentage of system RAM

The default cache size is 1/8 of the total memory.

-d *delay* The write-behind ("hold-off") delay applied to asynchronous writes, specified in seconds (default is 3 seconds). If the asynchronous write queue is empty, this delay is applied when the first block is placed in the queue. As a result, the queue is likely to contain many blocks before the driver is called to drain the queue.

-F Don't use priority float (i.e. always run at fixed priority). The default is to float to client-driven priority.

-f *ocb* The maximum number of *open control blocks* that may be open at one time in the filesystem (default is equal to the maximum number of processes **Proc** has been configured for. See "**Proc -p***procs*").

-H *heap_size* The size of heap to use; see "Heap size," below.

-i *inode_table_size*

The maximum number of distinct files that may be open at any time (default is twice the maximum number of processes).

-l *lock_table_size*

("el") The number of locks that can be simultaneously outstanding (default is equal to the maximum number of processes).

-O Process requests in time order, not priority order.

-P *percentage* Reserve this percentage of the cache for segmented LRU replacement. Default is 20% if the cache size is the default or larger. If the cache specified is smaller than the default cache, the reserved percentage is decreased. **-P0** effectively results in pure LRU caching.

-r *ramdisk_size*	The size of an optional ramdisk, in units of 1K. The minimum size is 4K; the maximum size is limited by available memory. There is no default size. A block special file called **/dev/ram** is created.
-t *threads*	The number of threads (default is 4, minimum is 1)
-T	Disable threaded operation in driver.
-v *level*	The verbosity level (default is 1; 0 disables verbosity).

Description:

The Filesystem Manager (**Fsys**) provides a standardized means of storing and accessing data on disk subsystems. **Fsys** is responsible for handling all requests to open, close, read, and write to files.

The **Fsys** manager lets you define *ramdisks*. A ramdisk is a block of memory that's accessed by the filesystem as if it were a disk drive. It can be effectively used for temporary files or to augment the cache by storing frequently used data, such as programs or tables. Only one ramdisk is supported by each Filesystem Manager process.

The ramdisk is defined as a block special file with the name **/dev/ram**. To define the root name for the filesystem on the ramdisk, you use the **mount** utility. But to use the ramdisk as a QNX filesystem, you must first **dinit** it. For example:

```
Fsys -r 1048
dinit /dev/ram
mount /dev/ram  /ram
```

Remember that since a ramdisk is entirely RAM-based, its contents are lost when the system is rebooted.

The maximum number of distinct files that may be open at any time may be less than the total number of open files, since files with links have only one entry and the same file opened by several processes at the same time has only one entry.

Heap size

Fsys requires data memory for the filesystem buffer cache (an in-memory copy of LRU disk sectors, since memory is much faster than disk) and for data structures such as open files, inodes, locks, name caching, and so on. It also stores summary bitmap information about free and used blocks on mounted filesystems.

Fsys and the **Fsys.*** disk drivers share a memory segment to give the drivers direct access to the filesystem buffer cache that **Fsys** manages (for no-copy physical data transfer, so a sector can be transferred directly between the disk and the cache without the need for intermediate buffering in the disk driver).

Under QNX 4, once a memory segment has been shared, it can't be resized. Thus, **Fsys** must initially allocate a maximal heap to itself before any **Fsys.*** drivers register. It knows the amount of cache, open files, and so on from the defaults or command-line overrides (**-c**, **-f**, **-i**, **-l**, **-t**, and **-C**). However, at this point it doesn't know the size of any attached disks, or the number of partitions or filesystems they may contain, or even the number of **Fsys.*** drivers that will attach to it.

So **Fsys** makes a guess as to those disk-related unknowns (a certain number of drivers, each with a certain number of disks and partitions, of a certain size). Historically it was a reasonable estimate, but with the increase in modern disk sizes, it's often too low.

If the **Fsys** heap runs out, the symptoms are primarily spurious ENFILE ("too many open files") or ENOMEM ("not enough memory") errors. Note that as the allocations are being performed from the pregrown **Fsys** heap (and can't be satisfied from the OS by growing the data segment), these errors bear no relationship to the actual amount of free memory in the system itself.

Whenever **Fsys** can't allocate memory for a filesystem object, it places an entry into the **traceinfo** log as follows:

00003024 internal heap exhaustion (nbytes=*d***/***t***) (object=***s***)**

If you see this message, the amount of memory estimated or preallocated to **Fsys** is insufficient for the actual size of the attached disks/filesystems. In this case, use the **Fsys -H** option to override or augment the heap size.

Version 4.24Y or later of **Fsys** supports the syntax **-H disk***x*, where *x* is the total size (in GB) of all disks drives in the system (e.g. **-Hdisk40** for 40 GB). This sets an amount (12 KB for every 1 GB of disk) to be added to the known memory requirements and estimations for drivers/partitions.

Otherwise, you can use the syntax **-H** *x* to set the amount of heap memory over and above the known requirements. In this case, the value must also include the memory needed for drivers, disks, and partitions, as well as the bitmap summary information for mounted filesystems. You can use the information in the **traceinfo** log to calculate a suitable value. The two numeric values logged are the size of the guess made for disk-related usage (*d*) and the total size of the preallocated heap (*t*). The difference is thus the amount of memory for known requirements, such as open files, inodes, and so on. What you're specifying with the **-H** parameter in this case overrides that first numeric value (*d*).

If you start **Fsys** from the boot image, don't forget to rebuild the image and copy it to **.boot** for the new **-H** value to take effect.

Driver shutdown

If you need to shut down a filesystem driver, follow these steps:

CAUTION: Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

1. Unmount (**umount**) all mounted partitions.

2. Unmount (**umount**) all partition tables that had been read via **umount** on the device. For example, if you've done: **mount -p /dev/hd0** you'll need to do: **umount /dev/hd0**.

3. Remove all devices owned by the driver you want to shut down — these are block special files that reside under **/dev**. For example: **rm /dev/hd0**.

When the block special files have been removed, the driver that handles them will go away.

Examples:

Start the filesystem manager with default settings:

```
Fsys &
```

Files:

Fsys closes its standard input, standard output, and standard error after initialization has been completed. Error messages during initialization will be written to standard error, except when **Fsys** isn't being run by **root**, in which case a message is written to standard output.

Fsys adopts block-special files under **/dev** and adopts other files under the filesystem mountpoints specified to the **mount** utility.

Exit status:

Fsys normally will not terminate. However, it is possible for it to exit if it encounters errors on startup.

1 Not being run by **root** *or* failed to allocate internal process table (memory)

2 Failed to allocate thread space (memory)

3 Failed to allocate cache (memory)

4	Failed to allocate file descriptions (memory)
5	Failed to allocate inodes (memory)
7	Failed to allocate lock table (memory)
8	Failed to allocate ramdisk (memory)
9	Failed during *init_fd()*
10	Failed to attach name
11	Failed to obtain driver proxy
12	Failed to obtain timer proxy
13	Failed to create timer
14	Failed to obtain major device number
15	Failed to allocate (reserve) heap (memory)

Caveats:

Although the **-a** option may speed up certain operations, its use, especially with a large delay (option **-d**), can seriously affect the reliability of the filesystem in the event of an unplanned shutdown, such as a power failure. You should use the **-a** option with extreme caution only after experimentation has shown that its use is necessary and only on a reliable system (possibly equipped with a UPS — uninterruptible power supply).

See also:

Blkfsys, **Dosfsys**, **Fsys.***, **Iso9660fsys**, **mount**, **Pipe**

Fsys.aha2scsi

Driver for Adaptec 6260/6360-based SCSI host adapters (QNX)

Syntax:

Fsys.aha2scsi [*general_options***] [fsys** *fsys_options***] [aha2scsi** *aha2scsi_options***]...**

Options:

general_options

If specified, *general_options* must precede any **fsys** or **aha2scsi** options:

-a *unit*=*type*

Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted; see also the Examples section):

Specify	Meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

© 2005, QNX Software Systems

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scan_mask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results from all units.

-d
Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *heads*[**,** *sectors*]
Report this many heads (and optionally sectors) to **Fsys** for hard disks (default is 64 heads). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

© 2005, QNX Software Systems

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on removable and tape devices.

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of activity before stopping direct drive (default: 0 — disabled)
-S	Try to always use scatter/gather.

aha2scsi_options

The *aha2scsi_options* control the driver's interface to the AHA 2 series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **aha2scsi** keyword before each controller's set of options.

-a *controller*	Use this controller address. Default is 340 (hex).
-b	Use 8-bit transfers.
-d	Disable disconnect.
-i *irq*	Assume that the controller is using this interrupt. Default is 11.

-r	Reset the controller and SCSI bus at initialization time.
-s *SCSI_ID*	Specify the SCSI ID used by the controller. Default 7.
-u *num*	Allow no more than this many SCSI units per controller.
-w	Do not use double-word transfers.

Description:

Fsys.aha2scsi is the driver for the Adaptec AIC-6260 and AIC-6360 based SCSI controllers (e.g. 1522). It detects emx16, aha 1520, and embedded 6260 and 6360 (motherboard) controllers. If the driver detects an embedded 6260 or 6360 and no command-line arguments exist, the factory defaults are used.

Controllers supported by this driver include, but are not necessarily limited to:

Manufacturer	Controller
Adaptec	AIC-6260
Adaptec	AIC-6360
Adaptec	AHA-1510A
Adaptec	AHA-1520A
Adaptec	AHA-1522A
Adaptec	AHA-1530P
Adaptec	AHA-1570
Creative Labs	Sound Blaster 16 SCSI

The driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit

Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively.

The Filesystem Manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

7 **optical**

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

CAUTION:

To boot from a unit connected to the controller, make sure the controller is configured for a 64-head BIOS or specify the driver's **fsys -h** option.

Examples:

Start the Adaptec 2 driver:

```
Fsys.aha2scsi
```

Register the direct-access unit as **/dev/hard***n* and the sequential-access unit as **/dev/tape***n*:

```
Fsys.aha2scsi fsys -n direct=hard -n seq=tape
```

Support a single AHA controller at address 0x334 and run quietly. Register direct-access units (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.aha2scsi -q fsys -n 0=scsi aha2scsi -a 334
```

Use sixteen 512-byte deblocking buffers for two CD-ROMs that each use 2048-byte sectors:

```
Fsys.aha2scsi fsys -B 16
```

In the above example, the argument for **-B** was calculated as follows:

$$16 = (2 \times (2048 \div 512)) + (2 \times (2048 \div 512))$$

Support all AHA 2 series controller and a CD-ROM that has a SCSI target ID of 4 (in case the CD-ROM isn't powered up when the driver starts, we've used the **-a** option):

```
Fsys.aha2scsi -a 4=CD
```

If you had two SCSI controllers and the CD-ROM was connected to the second one (determined by the driver's scan), you would modify the above command as follows:

```
Fsys.aha2scsi -a 1,4=CD
```

Files:

Fsys.aha2scsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.aha2scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named **hd***n*, where *n* is the physical unit number of the device. The base portion of this filename (**hd**) may be overridden by the **-n** option.

Exit status:

Fsys.aha2scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.aha2scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.aha4scsi

© 2005, QNX Software Systems

Driver for Adaptec 154x and compatible SCSI host adapters (QNX)

Syntax:

Fsys.aha4scsi [*general_options***] [fsys** *fsys_options***]**
[aha4scsi *aha4scsi_options***]...**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **aha4scsi** options:

-a *unit=type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted; see also the Examples section):

Specify	Meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
: Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

  The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

  $buf = (2 \times (sector_size \div 512))$

  The value that should be used for *num* will be the sum of the results from all units.

-d
: Allow concurrent I/O with other DMA-based drivers.

  Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *heads*[**,***sect*]
: Report this many heads (and optionally sectors per track) to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on removable and tape devices.

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of activity before stopping direct drive (default is 0 — disabled)
-S	Try to always use scatter/gather.

aha4scsi_options

The *aha4scsi_options* control the driver's interface to the AHA 4 series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **aha4scsi** keyword before each controller's set of options.

-a *controller*	Use this controller address. By default, the driver scans for all AHA 4 controllers. With this option, the driver doesn't perform a scan and uses only the controllers you specify.
-b *timeout*	Set the ISA/EISA bus busy timeout. By default, the driver queries the card for this value. For possible timeout values, see the *AHA-1540A/1542A Technical Reference* (available from Adaptec).

-d *dma_channel*

Assume that the controller uses this DMA channel for bus mastering. By default, the driver queries the controller for this value.

-f

Do a full scan for the installed SCSI units. By default, the driver uses the controller's built-in scan (see the **-u** option).

You should specify this option for any controller whose built-in scan may ignore removable media.

If you have removable hard disks and don't want to use the **-f** option with the 1540C controller, you must enable the controller's advanced configuration option "Support Removable Disks under BIOS as Fixed Disks."

If you specify **-f** and you use the **mount** command in your boot image or sysinit file, you may need to specify the **-w** (wait) option to **mount**. The full scan may take longer than **mount**'s default wait time.

-h *head*[**,***sect*]

Report this many heads (and optionally sectors per track) to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i *irq*

Assume that the controller is using this interrupt. By default, the driver queries the controller for this value.

-m *mboxes*

Support this many mailboxes (default is 16). Mailboxes are used for simultaneous access to multiple SCSI units.

-r

Reset the controller and SCSI bus at initialization time.

-t *speed*	Set the ISA/EISA bus transfer speed. For the possible values of *speed*, which is specified in hex, see the *AHA-1540A/1542A Technical Reference*.
-u *num*	Allow no more than this many SCSI units per controller. You may need to specify this option if you've also specified **-f**. By default, the driver sets the maximum number of units (devices) to the number of units found during the built-in scan. But with **-f**, the driver sets the maximum number of units to 4.
-x *on,off*	Set the on-and-off duration of the ISA/EISA bus. For possible values, see the *AHA-1540A/1542A Technical Reference*.

Description:

Fsys.aha4scsi is the driver for the Adaptec 154x and compatible SCSI disk controllers. The controller models support by this driver include, but are not necessarily limited to, these models:

Manufacturer	Controller
Adaptec	AHA-1540
Adaptec	AHA-1540A/1542A
Adaptec	AHA-1540B/1542B
Adaptec	AHA-1540C/1542C
Adaptec	AHA-1540CF/1542CF
Adaptec	AHA-1640
Adaptec	AHA-1740A/1742A (in 1540 emulation mode only)
Buslogic	BT-440C

continued...

Manufacturer	Controller
Buslogic	BT-542B
Buslogic	BT-542D
Buslogic	BT-545S
Buslogic	BT-545C
Buslogic	BT-440C
Buslogic	BT-445C
Buslogic	BT-646S
Buslogic	BT-747S
Buslogic	BT-946C

Fsys.aha4scsi scans for controllers in the command-line sequence given by the **-a** option. If no **-a** option is given, the driver scans for controllers at the following hex addresses, in order:

- 330
- 334
- 130
- 134
- 230
- 234

Controllers are numbered from 0 to *n*, in the order they're found.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively.

© 2005, QNX Software Systems

The Filesystem Manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

7 **optical**

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

CAUTION:

To boot from a unit connected to the controller, make sure the controller is configured for a 64-head BIOS or specify the driver's **fsys -h** option.

Examples:

Support all AHA 4 series controllers:

```
Fsys.aha4scsi
```

Register the direct-access unit as **/dev/hard***n* and the sequential-access unit as **/dev/tape***n*:

```
Fsys.aha4scsi fsys -n direct=hard -n seq=tape
```

Support a single AHA controller at address 0x334 and run quietly. Register direct-access units (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.aha4scsi -q fsys -n 0=scsi aha4scsi -a 334
```

Use sixteen 512-byte deblocking buffers for two CD-ROMs that each use 2048-byte sectors:

```
Fsys.aha4scsi fsys -B 16
```

In the above example, the argument for **-B** was calculated as follows:

$$16 = (2 \times (2048 \div 512)) + (2 \times (2048 \div 512))$$

Support all AHA 4 series controllers and a CD-ROM that has a SCSI target ID of 4 (in case the CD-ROM isn't powered up when the driver starts, we've used the **-a** option):

```
Fsys.aha4scsi -a 4=CD
```

If you had two SCSI controllers and the CD-ROM was connected to the second one (determined by the driver's scan), you would modify the above command as follows:

```
Fsys.aha4scsi -a 1,4=CD
```

Files:

Fsys.aha4scsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.aha4scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named **hd***n*, where *n* is the physical unit number of the device. The base portion of this filename (**hd**) may be overridden by the **-n** option.

Exit status:

Fsys.aha4scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.aha4scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

AHA-1540A/1542A Technical Reference (available from Adaptec)

Fsys.aha7scsi

Driver for Adaptec AIC-7770/7870/7880 based SCSI adapters

Syntax:

Fsys.aha7scsi [*general_options***] [fsys** *fsys_options***]**
[aha7scsi *aha7scsi_options***]**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **aha7scsi** options:

-a *unit*=*type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted; see also the Examples section):

Specify	Meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

© 2005, QNX Software Systems

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results from all units.

-d
Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *heads*[**,** *sectors*]
Report this many heads (and optionally sectors) to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

© 2005, QNX Software Systems

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on Removable & Tape

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of inactivity before stopping direct drive. (disabled)
-S	Try to always use Scatter/Gather.

aha7scsi_options

The *aha7scsi_options* control the driver's interface to the AHA 7 series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **aha7scsi** keyword before each controller's set of options.

-a *controller*	Use this controller address.
-d *mask*	Specify disconnect bitmask (default: **0xffff**).
-D *device_id*	Device ID of the controller.
-F *mask*	Specify fast synchronous transfers bitmask (default: **0xffff**).

-i *irq*	Assume that the controller is using this interrupt. By default, the driver queries the controller for this value.
-p *pci_index*	The PCI index of the controller in the machine, where *pci_index* is a value between 0 and the number of adapters.
-r	Do not reset the SCSI bus.
-S *mask*	Specify synchronous transfers bitmask (default: **0xffff**).
-s	Specify the SCSI id used by the controller.
-t *low*[**,***high*]	Set termination low, high (default is **1,1**).
-U *mask*	Ultra/Fast20 synchronous bitmask (default: **0xffff**).
-u *num*	Allow no more than this many SCSI units per controller.
-V *vendor_id*	Vendor ID of the controller.
-W *mask*	Specify wide transfers bitmask (default: **0xffff**).

Description:

Fsys.aha7scsi is the driver for SCSI adapters based on the Adaptec AIC-7770, AIC-7870 and AIC-7880 chips. Controllers supported by this driver include, but are not necessarily limited to:

Manufacturer	Controller
Adaptec	AIC-7770
Adaptec	AIC-7870

continued...

Manufacturer	Controller
Adaptec	AIC-7880
Adaptec	AHA-2740/2742
Adaptec	AHA-2740T/2742T
Adaptec	AHA-2740W/2742W
Adaptec	AHA-2840A/2842A
Adaptec	AHA-2940
Adaptec	AHA-2940W
Adaptec	AHA-3940

Fsys.aha7scsi scans for controllers in the command-line sequence given by the **-a** option. If no **-a** option is given, the driver scans for controllers at the following hex addresses, in order:

1c00
2c00
3c00
4c00
5c00
6c00
7c00
8c00
9c00
ac00
bc00
cc00
dc00
ec00
fc00

After the above addresses are scanned the BIOS is queried for PCI cards.

If you have problems with the PCI adapter make sure that you have an up to date version of the adapter BIOS as well as system BIOS.

Controllers are numbered from 0 to *n*, in the order they're found.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively.

The Filesystem Manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

7 **optical**

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

Examples:

Support an AHA 7 controller, list all connected devices:

```
Fsys.aha7scsi &
```

Support an AHA 7 PCI controller with a *pci_index* of 1, list all connected devices:

```
Fsys.aha7scsi aha7scsi -p1 &
```

Support an AHA 7 controller, list all connected devices, and register direct access devices as **/dev/hard***n* and sequential access devices as **/dev/tape***n*:

```
Fsys.aha7scsi fsys -n0=hard -n1=tape &
```

Support an AHA 7 controller at hex address 1c00, don't display list of connected devices. Register direct access devices (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.aha7scsi -q fsys -n0=scsi aha7scsi -a1c00 &
```

Files:

Fsys.aha7scsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.aha7scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named according to the device types in the form *typen*, where *type* is the abbreviated device type and *n* is the physical unit number of the device. The names used for the abbreviated device type may be overridden via the **-n** *type*=*n* option.

© 2005, QNX Software Systems

Exit status:

Fsys.aha7scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.aha7scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.aha8scsi

Driver for Adaptec common HIM SCSI adapters

Syntax:

Fsys.aha8scsi [*general_options*] [**fsys** *fsys_options*] [**aha8scsi** *aha8scsi_options*]

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **aha8scsi** options:

-a *unit*=*type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted; see also the Examples section):

Specify	Meaning
path,*target*,*lun*	fully specified path, target, and LUN
path,*target*	equivalent to *path*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

© 2005, QNX Software Systems

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as needed to uniquely identify the type. For example, to indicate type **1**, you could specify **se**. Note that strings are case-insensitive.

Not all the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask. (Default: all targets.)

-v Be verbose: display full information about SCSI units (devices) on startup.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
Use this many 512-byte buffers for blocking/deblocking. (Default: 8.) You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results from all units.

-d
Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *heads*[**,** *sectors*]
Report this many heads (and optionally sectors) to **Fsys** for hard disks. (Default: 64 heads, 32 sectors per track.) The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

© 2005, QNX Software Systems

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on Removable & Tape

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation. (Default: 127.)

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of inactivity before stopping direct drive. (Default: disabled.)
-S	Try to always use Scatter/Gather.

aha8scsi_options

The *aha8scsi_options* control the driver's interface to the AHA common HIM controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **aha8scsi** keyword before each controller's set of options.

The target bitmask is specified in hexadecimal (e.g. **0xffff**); each bit in the mask corresponds to a SCSI ID number. For example, **0xf001** would indicate SCSI IDs 15, 14, 13, 12, and 0.

-d *mask*	Specify disconnect bitmask. (Default: **0xffff**, i.e. allow disconnects.)

-D *device_id* Device ID of the controller. (Default: **0x0000**.) Used in conjunction with the *vendor_id* to generate the PCI ID of the Adaptec controller to be searched for in PCI space.

-F *mask* Specify fast synchronous transfers bitmask. (Default: **0xffff**, i.e. allow fast transfers.)

-i *irq* Assume that the controller is using this interrupt. By default, the driver queries the controller for this value.

-p *pci_index* The PCI index of the controller in the machine, where *pci_index* is a value between 0 and the number of adapters. (Default: 0.)

-S *mask* Specify synchronous transfers bitmask. (Default: **0xffff**, i.e. allow sync.)

-T *Timeout* When specified, this option will change the default CAM command Timeout value, in seconds. (Default: 120.) This option isn't intended for normal use.

-U *mask* Ultra/Fast20 synchronous bitmask. (Default: **0xffff**, i.e. allow Ultra.)

-u *num* Allow no more than this many SCSI units per controller. (Default: *MaxTargets* × *MaxLuns*.)

-V *vendor_id* Vendor ID of the controller. (Default: **0x9004**, Adaptec.) Used in conjunction with *device_id* to generate the PCI ID of the Adaptec controller to be configured.

-W *mask* Specify wide transfers bitmask (Default: **0xffff**, i.e. allow wide transfers.)

Description:

Fsys.aha8scsi is the driver for SCSI adapters based on the Adaptec AIC-7870, AIC-7880, and AIC-7890 chips. Controllers supported by this driver include at least the following:

Manufacturer	Controller
Adaptec	AIC-7870
Adaptec	AIC-7880
Adaptec	AIC-7890
Adaptec	AHA-2940 series
Adaptec	AHA-3940 series

If you have problems with the PCI adapter make sure that you have an up-to-date version of the adapter BIOS as well as system BIOS.

Controllers are numbered from 0 to *n*, in the order they're found.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively.

The Filesystem Manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

```
7    optical
```

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

Examples:

Support an AHA 8 controller, list all connected devices:

```
Fsys.aha8scsi &
```

Support an AHA 8 PCI controller with a *pci_index* of 1, list all connected devices:

```
Fsys.aha8scsi aha8scsi -p1 &
```

Support an AHA 8 controller, list all connected devices, and register direct access devices as **/dev/hard***n* and sequential access devices as **/dev/tape***n*:

```
Fsys.aha8scsi fsys -n0=hard -n1=tape &
```

Support an AHA 7 controller at hex address 1c00, don't display list of connected devices. Register direct access devices (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.aha8scsi -q fsys -n0=scsi aha8scsi -a1c00 &
```

Files:

Fsys.aha8scsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.aha8scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named according to the device types in the form *typen*, where *type* is the abbreviated device type and *n* is the physical unit number of the device. The names used for the abbreviated device type may be overridden via the **-n** *type*=*n* option.

Exit status:

Fsys.aha8scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.aha8scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.amdscsi

© 2005, QNX Software Systems

Driver for AMD 53c974/79C974 PCI SCSI Host adapters

Syntax:

Fsys.amdscsi [*general_options***...] [fsys** *fsys_options***...] [amdscsi** *amdscsi_options***...]...**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **amdscsi** options:

-a *unit*=*type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted; see also the Examples section):

Specify	Meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

© 2005, QNX Software Systems

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results from all units.

-d
Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *head*[**,***sect*]
Report this many heads (and optionally sectors) to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on removable and tape devices.

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of inactivity before stopping direct drive (default 0 == disabled)
-S	Try to always use scatter/gather.

amdscsi_options

The *amdscsi_options* control the driver's interface to the AMD 53c974 series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **amdscsi** keyword before each controller's set of options.

-d	Disable disconnect.
-f	Enable Fast SCSI.
-p *pci_index*	The PCI index of the controller in the machine, where *pci_index* is a value between 0 and the number of adapters.
-r	Do not reset controller at initialization time.

-s *SCSI_ID*	The *SCSI_ID* used by the controller (7).
-S	Disable synchronous transfers.
-u *num*	Maximum number of SCSI units.

Description:

Fsys.amdscsi is the driver for AMD 53c974 and AMD 79c974 SCSI disk controllers.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively.

Fsys.amdscsi will autodetect all AMD 53c974 and AMD 79c974 adapters by default, therefore you *must* specify the **-p** *pci_index* option to the driver if you want a specific SCSI adapter to be used.

The Filesystem Manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

7 **optical**

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0**

and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

Examples:

Support an AMD 53c974 controller, list all connected devices:

```
Fsys.amdscsi &
```

Support an AMD 53c974 controller with a *pci_index* of 1, list all connected devices:

```
Fsys.amdscsi amdscsi -p1 &
```

Support an AMD 53c974 controller, list all connected devices, and register direct access devices as **/dev/hard***n* and sequential access devices as **/dev/tape***n*:

```
Fsys.amdscsi fsys -n0=hard -n1=tape &
```

Files:

Fsys.amdscsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.amdscsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named **hd***n*, where *n* is the physical unit number of the device. The base portion of this filename (**hd**) may be overridden by the **-n** option.

Exit status:

Fsys.amdscsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.amdscsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.ata

© 2005, QNX Software Systems

Driver for ATA fixed-disk interface (QNX)

Syntax:

Fsys.ata [*options***...]**

Options:

-a *controller*	Assume the controller is at this address (default is 1f0 hex).
-B	Ignore BIOS settings for the drive.
-c *drivespec*	Assume these parameters for the drive. You specify these in the following format: *heads*,*cylinders*,*sectors*[,*disknum*] You must include *heads*, *cylinders*, and *sectors*. If you omit *disknum*, unit 0 is assumed. (Default: autodetect)
-f *fsysname*	Registered name of filesystem process to attach to. (**qnx/fsys32**)
-i *irq*	Assume the controller is using this interrupt (default is 14).
-m *maxsize*	Allow no more than this many blocks to be transferred from the disk to memory in one operation (default is 127).
-M *multi_limit*	Maximum number of blocks per interrupt. (Default: maximum supported by the drive.)
-n *device_name*	Assign this name to the block special device associated with the hard disk. The default is **hd**, which yields a device name of **/dev/hd***n*, where *n* is a unit number assigned by **Fsys**.

-N *version_name*

Assign this unique version name to this invocation of the driver. The name can be from 1 to 11 characters in length. (Default: IDE)

-p Program the drive to use the highest I/O mode it is capable of.

-R *nretries* Number of times to retry bad blocks (default 1)

-v Verbose — report information about drives on startup.

-V Do not scan for CMD-640 VL interface.

Description:

Fsys.ata is the driver for the AT-Attachment (ATA) disk drive interface. This technology integrates the traditionally host-based controller electronics within the drive itself.

If . . .	Then use . . .
you have an ATAPI device (e.g. CD-ROM)	**Fsys.eide**
you have a removable hard drive	**Fsys.eide**
you have only a hard drive but no shortage of memory	**Fsys.eide**
you are low on memory, and have only a hard drive with LBA enabled	**Fsys.ata**
you are low on memory, and have only a hard drive without LBA enabled	**Fsys.ide**

continued. . .

If . . .	Then use . . .
you are booting off the drive and your BIOS doesn't support LBA	**Fsys.ide** or **Fsys.ata -L**
you are booting off the drive and the BIOS is set for LBA	**Fsys.ata**
you are booting off the drive and the BIOS is set for CHS, and the drive supports LBA	**Fsys.ide** or **Fsys.ata -L**

Note: If you are installing multiple OSs on the drive, make sure all OSs use a compatible mode. For example, if your drive is **>**=528Mb and DOS will also be installed on the drive, the driver should be configured to use LBA.

In order to obtain drive parameters, **Fsys.ata** examines values established by the BIOS of your computer. If necessary, you can override these default parameters with the **-c** option.

Fsys.ata will use LBA if the drive supports it. If the **-p** option is specified, the driver will use the highest PIO mode supported by the drive.

LBA is required if the drive is greater than 528M and DOS has been installed on the drive.

To define the largest single read or write the driver will support, use the **-m** *maxsize* option; the default value for this option is 127.

To help prevent you from unintentionally starting the same driver twice, **Fsys** rejects the invocation of any driver that has the same version name as a previously mounted driver. So if you wish to run multiple copies of **Fsys.ata**, you must use the **-N** *version_name* option to assign a unique version name to the driver. (You can display the version name of a driver with the **sin ver** command.) You

should also use the **-n** *device_name* option to specify a unique device name, since another driver may have already created the default device name.

Caveats:

Some IDE drives don't match the draft standard that **Fsys.ide** assumes (*AT Attachment Interface for Disk Drives*, Rev 4, March 17, 1993). If a drive won't function with **Fsys.ide**, please inform our technical support group. We'll need the make, model, and year of purchase of your drive.

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.eide

© 2005, QNX Software Systems

Driver for ATA/IDE disk interface and ATAPI CD-ROM interface

Syntax:

Fsys.eide [*general_options***] [fsys** *fsys_options***...]**
[eide *eide_options***...]**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **eide** options:

-q Be quiet: don't display any information on startup.

-v Be verbose.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num* Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results for all units.

-d Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur

concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *head*[**,***sect*] Report this many heads (and optionally, sectors) to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on Removable and Tape.

-m *num* Transfer no more than this many 512-byte sectors from the controller to memory in one operation (default is 128).

-M *num* Specify the maximum number of drives supported.

-n *type***=***name* For the given unit type, use the specified name. For example, to have hard drives use **/dev/eide***n*, you would specify **-n 0=eide**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**

continued...

© 2005, QNX Software Systems

type	String equivalent	Default *name*
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

You can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name* Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.

-r Reassign bad blocks automatically.

-R Do not try to reserve the unit while mounted/open.

-s *mins* Minutes of inactivity before stopping direct drive. (disabled by default)

-S Try to always use scatter/gather.

-t *num* Specify maximum concurrent I/O threads.

eide_options

The *eide_options* control the driver's interface to the EIDE controller. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **eide** keyword before each controller's set of options.

-a *address* The hex address of the controller. Default 1f0/170.

-b Do not use BIOS for drive parameters. BIOS is used by default.

-c *heads,cyl,sect*[,*disknum*]

Specify drive geometry. Note that *disknum* is the disk number on the controller, 0 indicating the first disk.

-e Use enhanced PIO modes. PIO modes are not used by default.

-i *irq* The interrupt used by the controller. Default 14/15.

-l *lba* Use CHS instead of LBA. LBA used by default.

-p Don't scan for PCI interfaces.

-S Don't scan for slave device (default is to scan).

-v Don't scan for CMD-640 VL interface.

Description:

Fsys.eide is the driver for IDE (Integrated Drive Electronics) EIDE (Enhanced IDE) ATA (AT Attachment) hard disks and ATAPI (ATA Packet Interface) CD-ROM interfaces.

If	Then use
you have an ATAPI device (e.g. CD-ROM)	**Fsys.eide**
you have a removable hard drive	**Fsys.eide**
you have only a hard drive but no shortage of memory	**Fsys.eide**
you are low on memory, and have only a hard drive with LBA enabled	**Fsys.ata**

continued...

If	Then use
you are low on memory, and have only a hard drive without LBA enabled	**Fsys.ide**
you are booting off the drive and your BIOS doesn't support LBA	**Fsys.ide** or **Fsys.ata -L**
you are booting off the drive and the BIOS is set for LBA	**Fsys.ata**
you are booting off the drive and the BIOS is set for CHS, and the drive supports LBA	**Fsys.ide** or **Fsys.ata -L**

Note: If you are installing multiple OSs on the drive, make sure all OSs use a compatible mode. For example, if your drive is **>**=528Mb and DOS will also be installed on the drive, the driver should be configured to use LBA.

Fsys.eide autodetects interfaces at address 0x1f0, interrupt 14 and address 0x170 interrupt 15 by default. If you have an interface at a different address/interrupt specify them to the driver.

Fsys.eide will use LBA (Logical Block Addressing) modes if the drive supports them.

If the **-l** option is specified, LBA is not used and the device is programmed to Cylinder-Head-Sector (CHS) mode. If the **-e** option is specified, enhanced PIO modes are used.

Examples:

Support an IDE controller, list all connected devices:

```
Fsys.eide &
```

Support an IDE controller at hex address 170, interrupt 15 and don't display list of connected devices. Register direct access devices (e.g. hard disks) as **/dev/eide#**:

```
Fsys.eide -q fsys -n0=eide eide -a170 -i15 &
```

Files:

Fsys.eide closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.eide will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named **hd***n*, where *n* is the physical unit number of the device. The base portion of this filename (**hd**) may be overridden by the **-n** option.

Exit status:

Fsys.eide will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.eide** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

© 2005, QNX Software Systems

See also:

Blkfsys, **Fsys**, **Fsys.***

© 2005, QNX Software Systems

Fsys.floppy

Driver for floppy disk controller (QNX)

Syntax:

Fsys.floppy [*options***...]**

Options:

-a *controller* The controller address (default is 3f0 hex).

-d *channel* Set DMA channel (default is 2).

-D *level* Print diagnostic information:

- **1** show errors
- **2** show results of all activity
- **3** show controller level information

-i *interrupt* The interrupt level used by the controller (default is 6).

-m *maxsize* The maximum number of blocks that will be transferred from the diskette to memory in one operation (default is 36).

-n *device_name* The name of the block special device associated with the floppy. The default is **fd**, which yields a device name of **/dev/fd***n*, where *n* is a unit number assigned by **Fsys**.

-N *version_name*

A unique version name to be assigned to this invocation of the driver. The name can be from 1 to 11 characters in length.

-s *id* Bus type (Ziatech hardware only)

-V *file* Direct diagnostics into the named file rather than to the standard output.

© 2005, QNX Software Systems

Description:

Fsys.floppy is the driver for IBM PC/AT/PS2-compatible floppy disk controllers. These controllers are based on the NEC 765 FDC, the INTEL 8237 DMA controller, and some extra control hardware defined by IBM for the PC.

Since these controllers use the 8237 DMA controller to transfer data between memory and the device, the driver will attempt to allocate up to *maxsize* × 512 bytes upon startup.

Ideally, *maxsize* should be the number of sectors in a cylinder, allowing the driver to read an entire cylinder in a single request. The default is 36, which is the number of sectors in a cylinder on a 1.44M drive.

This driver automatically changes the drive parameters to match the disk in the drive (e.g. 360K or 1.2M; 720K, 1.4M or 2.8M). It does this by reading the first block of the disk, on which the QNX **fdformat** has written the information used to set the drive parameters (heads, sectors/track, cylinders). When using disks from UNIX systems, e.g. **tar** archives on floppy disks, you must disable this feature of the driver by using the **lockfd** utility. The feature may be re-enabled via the same utility.

When running multiple copies of **Fsys.floppy**, you must use the **-N** *version_name* option to assign a unique version name to the driver. To help prevent you from unintentionally starting the same driver twice, **Fsys** rejects the invocation of any driver that has the same version name as another previously mounted driver. (You can display the version name of a driver with the **sin ver** command.) You should also use the **-n** *device_name* option to specify a unique device name, since another driver may have already created the default device name.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA will usually not perform I/O concurrently, unless one or both drivers has been configured to allow

this by command-line option. (**Fsys.floppy** cannot be configured this way.)

CAUTION: This driver is sensitive to the drive type specified in the non-volatile setup information in AT and PS/2 computers. Specifying the wrong type of drive during this setup might result in sporadic errors.

The driver recognizes and mounts the proper parameters for diskettes from QNX 4, QNX 2.1, and several versions of DOS. If you need to specify additional parameters to the driver, you use the *qnx_ioctl()* C function.

Examples:

Start **Fsys.floppy** using the configuration information in the CMOS setup data for drive information:

```
Fsys.floppy &
```

Files:

Fsys.floppy closes its standard input, standard output and standard error immediately after performing its startup initialization.

Fsys.floppy will cause **Fsys** to adopt block special files under **/dev** for each floppy device being serviced by the driver. These filenames are normally **/dev/fd***n* where *n* is the drive number. The base portion of the filename (**fd**) can be overridden by using the **-n** option.

Exit status:

Fsys.floppy will terminate only in response to a signal or errors encountered during its startup initialization.

0 No floppy drives were detected in the system, or **Fsys.floppy** was started in the foreground and **Fsys.floppy** successfully forked a copy of itself.

>0 An error occurred during initialization. An error message will have been written to standard error.

Caveats:

You can specify the controller address (via option **-a**), but your floppy disk controller hardware must match the IBM specification, which involves more than the NEC 765 FDC.

The floppy controller interface and timing requirements are poorly specified; variations from machine to machine make this unreliable hardware.

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **dinit**, **fdformat**, **Fsys**, **Fsys.***, **lockfd**, **pax**, **tar**

Fsys.ide

Driver for IDE fixed-disk interface (QNX)

Syntax:

Fsys.ide [*options***]...**

Options:

-a *controller*	Assume the controller is at this address (default is 1f0 hex).
-B	Ignore BIOS settings for the drive.
-c *drivespec*	Assume these parameters for the drive. You specify these in the following format: *heads*,*cylinders*,*sectors*[,*disknum*] You must include *heads*, *cylinders*, and *sectors*. If you omit *disknum*, unit 0 is assumed. (Default: autodetect)
-f *fsys_name*	Registered name of filesystem process to attach to (default: **qnx/fsys32**)
-i *irq*	Assume the controller is using this interrupt (default is 14).
-L	Do not use LBA.
-m *maxsize*	Allow no more than this many blocks to be transferred from the disk to memory in one operation (default is 127).
-M *multi_limit*	Maximum number of blocks per interrupt. (Default: maximum supported by the drive.)
-n *device_name*	Assign this name to the block special device associated with the hard disk. The default is **hd**, which yields a device name of **/dev/hd***n*, where *n* is a unit number assigned by **Fsys**.

-N *version_name*
: Assign this unique version name to this invocation of the driver. The name can be from 1 to 11 characters in length. (Default: IDE)

-p
: Program drive to use whatever IO mode it is capable of, regardless of how it has been configured.

-R *nretries*
: Number of times to retry bad blocks (default: 1)

-v
: Verbose. Report information about drives.

-V
: Do not scan for CMD-640 VL interface.

Description:

Fsys.ide is the driver for the Integrated Drive Electronics (IDE) disk drive interface. The IDE technology integrates the traditionally host-based controller electronics within the drive itself.

If	Then use
you have an ATAPI device (e.g. CD-ROM)	**Fsys.eide**
you have a removable hard drive	**Fsys.eide**
you have only a hard drive but no shortage of memory	**Fsys.eide**
you are low on memory, and have only a hard drive with LBA enabled	**Fsys.ata**
you are low on memory, and have only a hard drive without LBA enabled	**Fsys.ide**

continued...

If	Then use
you are booting off the drive and your BIOS doesn't support LBA	**Fsys.ide** or **Fsys.ata -L**
you are booting off the drive and the BIOS is set for LBA	**Fsys.ata**
you are booting off the drive and the BIOS is set for CHS, and the drive supports LBA	**Fsys.ide** or **Fsys.ata -L**

Note: If you are installing multiple OSs on the drive, make sure all OSs use a compatible mode. For example, if your drive is **>**=528M and DOS will also be installed on the drive, the driver should be configured to use LBA.

Manufacturers of IDE drives have collectively endorsed a standards process that defines a common command and cabling specification. The standard they've developed is now a "working draft" of the American National Standard of Accredited Standards Committee X3. For drives that vary from our interpretation of the IBM PC/AT specification and won't work with **Fsys.ata**, use **Fsys.ide**.

In order to obtain drive parameters, **Fsys.ide** examines values established by the BIOS of your computer. If necessary, you can override these default parameters with the **-c** option.

To define the largest single read or write the driver will support, use the **-m** *maxsize* option; the default value for this option is 127.

To help prevent you from unintentionally starting the same driver twice, **Fsys** rejects the invocation of any driver that has the same version name as a previously mounted driver. So if you wish to run multiple copies of **Fsys.ide**, you must use the **-N** *version_name* option to assign a unique version name to the driver. (You can display the version name of a driver with the **sin ver** command.) You should also use the **-n** *device_name* option to specify a unique device

name, since another driver may have already created the default device name.

The **Fsys.ide** driver does not support CD-ROM drives. See **Fsys.eide**.

Caveats:

Some IDE drives don't match the draft standard that **Fsys.ide** assumes (*AT Attachment Interface for Disk Drives*, Rev 4, March 17, 1993). If a drive won't function with **Fsys.ide**, please inform our technical support group. We'll need the make, model, and year of purchase of your drive.

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.mitsumi

Driver for Mitsumi LU-005 and FX-001 CD-ROM

Syntax:

Fsys.mitsumi [-a *controller***] [-i** *interrupt***] [-m** *maxsize***] [-n** *name***]**

Options:

-a *controller*	The controller address (default is 300 hex).
-i *interrupt*	The interrupt level used by the controller (default is 10).
-m *maxsize*	The maximum number of blocks that can be transferred from the disk to memory in one operation (default is 127, max 127).
-n *name*	The name of the block special device associated with the hard drive. The default is **cd** (e.g. **/dev/cd0**)

Description:

Fsys.mitsumi supports the non-IDE (non-ATAPI) Mitsumi internal CD-ROM drive models LU-005 and FX-001. The FX-001 unit is run in "double-spin" mode.

This driver supports only audio CDs and data CDs recorded in mode 1 (most CD-ROMs are mode 1). It does not have support for mode 2 or multi-session CDs (photo CDs).

Examples:

Start the **Fsys.mitsumi** driver to work with hardware in its default configuration:

```
Fsys.mitsumi &
```

© 2005, QNX Software Systems

Files:

This driver closes its standard input, standard output and standard error after startup. If an error occurs during startup, a diagnostic message is written to the standard error and **Fsys.mitsumi** exits with a non-zero exit status.

Fsys.mitsumi causes **Fsys** to register a block special file in **/dev** directory. This name will normally be **/dev/cd***n*, but this may be changed by specifying a different name to the **-n** option.

Exit status:

>0 An error occurred during startup, or **Fsys.mitsumi** was killed via a signal.

0 **Fsys.mitsumi** was not started in the background, and successfully forked itself leaving a copy running in the background and the original returning a 0 exit status.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.ncr8scsi

Driver for the NCR 53c810/815/820/825 PCI SCSI Host adapter — Obsolete

Description:

Since QNX 4.25 Patch D, the **Fsys.ncr8scsi** driver has been replaced by **Fsys.sym8scsi**.

Fsys.ps2scsi

© 2005, QNX Software Systems

Driver for IBM PS/2 SCSI host adapter (QNX)

Syntax:

Fsys.ps2scsi [*general_options***] [fsys** *fsys_options***]**
[ps2scsi *ps2scsi_options***]...**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **ps2** options:

-a *unit=type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted; see also the Examples section):

Specify	meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,0
target	equivalent to 0,*target*,0

In the *type* argument, you can specify one of the following SCSI device types:

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
: Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

  The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

  $buf = (2 \times (sector_size \div 512))$

  The value that should be used for *num* will be the sum of the results for all units.

-d
: Allow concurrent I/O with other DMA-based drivers.

  Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *heads*[**,***sect*]
: Report this many heads (and optionally sectors) to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking when open on removable and tape devices.

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-N *name* Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access**(e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-R Disable reserving the SCSI unit when the unit is open.

-r Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table.

Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.

-S Try to always use scatter/gather.

-s *mins* Minutes of inactivity before stopping direct drive (default: disabled)

ps2scsi_options

The *ps2scsi_options* control the driver's interface to the PS/2 SCSI controller. Remember to specify the **ps2scsi** keyword before these options.

-a *controller* Use this controller address.

-i *irq* Assume the controller is using this interrupt. By default, the driver queries the controller for this value.

-l Display (R or W) in PS2 LCD window.

-u *num* Allow no more than this many SCSI units per controller (default is 4).

Description:

Fsys.ps2scsi is the driver for the IBM PS/2 SCSI controller.

Fsys.ps2scsi scans for a controller in the command-line sequence given by the **-a** option. If no **-a** option is given, the driver scans for a controller at the following hex addresses, in order:

3540
3548
3550
3558
3560
3568
3570
3578

The driver then scans for installed units. All targets are scanned (7 to 0). The driver numbers the units (starting at 0) and registers them with **Fsys** in the order it finds them.

The Filesystem Manager supports only the following types of SCSI units:

Type	Description
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
4	**WORM** (write once/read multiple)
5	**CD-ROM**
7	**optical**

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

CAUTION: To boot from a unit connected to the controller, make sure the controller is configured for a 64-head BIOS entry or specify the driver's **fsys -h** option.

Examples:

Support a PS/2 SCSI controller:

```
Fsys.ps2scsi &
```

Register the direct access unit as **/dev/hard***n* and the sequential access unit as **/dev/tape***n*:

```
Fsys.ps2scsi fsys -n direct=hard -n seq=tape &
```

Support a controller at address 0x3540 and run quietly. Register direct access units (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.ps2scsi -q fsys -n 0=scsi ps2scsi -a 3540 &
```

Support a PS/2 SCSI controller with 6 SCSI units and don't display the list of connected devices. Register direct access units (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.ps2scsi -q fsys -n 0=scsi ps2scsi -u 6 &
```

Use sixteen 512-byte deblocking buffers for two CD-ROMs that each use 2048-byte sectors:

```
Fsys.ps2scsi fsys -B 16 &
```

In the above example, the argument for **-B** was calculated as follows:

$$16 = (2 \times (2048 \div 512)) + (2 \times (2048 \div 512))$$

Files:

Fsys.ps2scsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.ps2scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named according to the device types in the form *typen*, where *type* is the abbreviated device type and *n* is the physical unit number of the device. The names used for the abbreviated device type may be overridden via the **-n** *type*=*n* option.

Exit status:

Fsys.ps2scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.ps2scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.sym8scsi

© 2005, QNX Software Systems

Driver for Symbios/NCR 53C8xx SCSI adapters (QNX)

Syntax:

Fsys.sym8scsi [*general_options***...] [fsys** *fsys_options***...]**
[sym8scsi *sym8scsi_options***...]...**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **sym8scsi** options:

-a *unit*=*type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted):

Specify	Meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

© 2005, QNX Software Systems

Description:

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results for all units.

-d
Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *num*[**,***sect*]
Report this many heads/sectors to **Fsys** for hard disks (default is 64 heads, 32 sectors). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking while open on removable and tape devices.

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of inactivity before stopping direct drive. (disabled)
-S	Try to always use Scatter/Gather.

sym8scsi_options

The *sym8scsi_options* control the driver's interface to the Symbios/NCR controllers.

-c *ChipSet*	When specified, this tells the driver which chipset feature list to use. If the *DeviceId* isn't specified, then this will be the chipset used during the HBA detection. (Default: 810)

Chipset	Device
810	53C810 (*VendorId*:*DeviceId* – **0x1000:0x0001**)
811	53C811 (*VendorId*:*DeviceID* – **0x1000:0x0005**)
815	53C815 (*VendorId*:*DeviceID* – **0x1000:0x0004**)
820	53C820 (*VendorId*:*DeviceID* – **0x1000:0x0002**)
825	53C825 (*VendorId*:*DeviceID* – **0x1000:0x0003**)
860	53C860 (*VendorId*:*DeviceID* – **0x1000:0x0006**)
875	53C875 (*VendorId*:*DeviceID* – **0x1000:0x000F**)
885	53C885 (*VendorId*:*DeviceID* – **0x1000:0x000D**)
895	53C895 (*VendorId*:*DeviceID* – **0x1000:0x000C**)
896	53C896 (*VendorId*:*DeviceID* – **0x1000:0x000B**)

-p *PciIndex* PCI index to use (Default: 0).
Forces PCI detection of the driver to this *PciIndex*. All PCI cards with the same PCI IDs have a *PciIndex* number. This allows you to specify the card you want. Use in association with **show_pci**. Normally, you'd need to specify **-p** only if you have more than one Symbios/NCR 53C8xx SCSI card in your system.

-u *MaxUnits* Maximum number of SCSI units to reserve Memory for. (Default: **(***MaxTargets* ***** *MaxLuns***)**)

© 2005, QNX Software Systems

-a *IoBase* When this is specified, the driver will scan for HBA with a matching *IoBase*.

-i *Irq* When this is specified, the driver will attach to this interrupt, regardless of PCI configuration information.

CAUTION: Be careful not to force **sym8scsi** to attach to an incorrect interrupt number — interrupts enabled from the card can cause spurious interrupts to another system driver.

-V *VendorId* Specifies chipset vendor.

-D *DeviceID* Used in conjunction with *ChipSet*. This will allow the driver to attempt to attach to a device with this PCI identifier, loading the feature set with that of the specified chipset.

CAUTION: Use this option with extreme caution — it's not intended for normal use.

-r When specified, no SCSI bus RESET will be done at start up.

-T *TimeOut* When specified, this will change the default CAM command timeout value. (Default: 32)

The **-T** option isn't intended for normal use.

-A *TargetMask* When this is set, targets who correspond to the *TargetMask* won't be allowed to abort commands. (Default: 0)

-S *TargetMask* Allow Sync on these Targets. (Default: supported == allowed)

-W *TargetMask* Allow Wide on these Targets. (Default: supported == allowed)

-x *TargetMask*	Allow Ultra2 on these Targets. (Default: supported == allowed)
-U *TargetMask*	Allow Ultra on these Targets. (Default: supported == allowed)
-F *TargetMask*	Allow Fast on these Targets. (Default: supported == allowed)
-v	Verbose output.

Fsys.sym8scsi is the driver for the Symbios/NCR 53c8xx SCSI adapters. The driver automatically scans the machine for controllers. Controllers are numbered from 0 to *n*, in the order they're found.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 6) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 6). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively. The **Fsys.sym8scsi** driver always sets the hosts SCSI ID to 7.

The filesystem manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

7 **optical**

This driver currently does not support synchronous or wide data transfers.

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

CAUTION:

To boot from a unit connected to the controller, make sure the controller is configured for a 64-head BIOS entry, or specify the driver's **fsys -h** option.

Examples:

Start all cards in the system:

```
Fsys -c512K &
Fsys.sym8scsi &
```

Start two cards in the system:

```
Fsys -c512K &
./Fsys.sym8scsi sym8scsi -c875 -p0 -v &
./Fsys.sym8scsi sym8scsi -c875 -p1 &
```

Set interrupts:

```
Fsys -c512K &
./Fsys.sym8scsi sym8scsi -c885 -i9 -p0 -v &
```

Disable LUN scan and start the second 53C860 device:

```
Fsys -c512K &
./Fsys.sym8scsi -L sym8scsi -c860 -p1 &
```

Files:

Fsys.sym8scsi closes its standard input, standard output, and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.sym8scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named according to the device types in the form *typen*, where *type* is the abbreviated device type and *n* is the physical unit number of the device. The names used for the abbreviated device type may be overridden via the **-n** *type*=*n* option.

Exit status:

Fsys.sym8scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.sym8scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.us14scsi

© 2005, QNX Software Systems

Driver for Ultrastor US-14F/34F SCSI Adapter

Syntax:

Fsys.us14scsi [*general_options***...] [fsys** *fsys_options***...] [us14scsi** *us14scsi_options***...]...**

Options:

general_options

If specified, the *general_options* must precede any **fsys** or **us14scsi** options:

-a *unit*=*type* Add this unit to the driver's device list. You should find this option handy in situations where a SCSI device might not be powered up when the driver performs its scan. Without **-a**, the only devices the driver will support are those that respond during the scan. You can specify multiple **-a** options.

The *unit* can be one of the following combinations (the number of commas affects how the unit information is interpreted):

Specify	Meaning
controller,*target*,*lun*	fully specified controller, target, and LUN
controller,*target*	equivalent to *controller*,*target*,**0**
target	equivalent to **0**,*target*,**0**

In the *type* argument, you can specify one of the following SCSI device types:

type	String equivalent
0	**direct-access** (e.g. hard disk)
1	**sequential-access** (e.g. tape)
2	**printer**
3	**processor**
4	**WORM**
5	**CD-ROM**
6	**scanner**
7	**optical**
8	**medium_changer**
9	**communications**

To specify the *type*, you can use either the number or its string equivalent. When specifying a string, you need include only as many characters as are necessary to uniquely identify the type. For example to indicate type **1**, you could specify **se**. Note that strings are case insensitive.

Not all of the above types are currently supported by the Filesystem Manager. See the **-n** option.

-L Disable automatic scan for LUNs.

-q Be quiet: don't display any information on startup.

-S *scanmask* Target scan bitmask (default: **0xffff**).

-v Be verbose: display full information about SCSI units (devices) on startup.

fsys_options

The *fsys_options* control the driver's interface to **Fsys**. If specified, they must follow the **fsys** keyword:

-B *num*
Use this many 512-byte buffers for blocking/deblocking (default is 8). You need to specify this option if you have one or more units that don't use 512-byte sectors. CD-ROM units, for example, may use 2048-byte sectors.

The value of *num* is the sum of all the buffers required for SCSI units that use sectors larger than 512 bytes. To obtain the buffer required for a unit, you must use the following formula:

$buf = (2 \times (sector_size \div 512))$

The value that should be used for *num* will be the sum of the results for all units.

-d
Allow concurrent I/O with other DMA-based drivers.

Some ISA motherboards have problems when on-board DMA and bus-mastering DMA occur concurrently. To avoid those problems, **Fsys** drivers which use DMA perform I/O concurrently only if the **-d** option is specified to either or both drivers. In general, you shouldn't use **-d** on an ISA machine if the floppy is going to be used at the same time as the SCSI hard disk. This option may be used safely on MCA and EISA machines.

-h *heads*[**,***sect*]
Report this many heads/sectors to **Fsys** for hard disks (default is 64 heads, 32 sectors per track). The QNX filesystem doesn't need this information for normal operation. The information is needed only to let **fdisk** write the correct boot cylinder for booting.

-i Ignore partition table when determining heads/sectors.

-l ("el") Disable locking when open on removable or tape devices.

-m *num* Transfer no more than this many 512-byte sectors from the SCSI controller to memory in one operation (default is 127).

-M *num* Allow no more than this many units to register with **Fsys** (default is the number of SCSI units available).

-n *type*=*name* For the given SCSI unit type, use the specified name. For example, to have hard drives use **/dev/scsi***n*, you would specify **-n 0=scsi**. You can specify multiple **-n** options.

By default, the driver assigns names as follows:

type	String equivalent	Default *name*
0	**direct-access** (e.g. hard disk)	**hd**
1	**sequential-access** (e.g. tape)	**tp**
4	**WORM**	**wo**
5	**CD-ROM**	**cd**
7	**optical**	**mo**

As in the **-a** general option, you can specify the *type* using either the number or its string equivalent.

To have the driver ignore a type of device, specify an empty setting. For example, to ignore hard disks, specify **-n 0**.

-N *name*	Assign this unique version name to this invocation of the driver. The name can range from 1 to 8 characters in length.
-r	Reassign bad blocks automatically. If a bad block is found on a drive, a "**REASSIGN BLOCKS**" command will be given to the controller, causing the drive to map the block through a defect-mapping table. Use this option with care since bad blocks could be erroneously reported by the device if the SCSI bus isn't properly terminated. If this occurs and you've specified **-r**, the defect-mapping table will be filled unnecessarily.
-R	Disable reserving the SCSI unit when the unit is open.
-s *mins*	Minutes of inactivity before stopping direct drive (default: disabled)
-S	Try to always use scatter/gather.

us14scsi_options

The *us14scsi_options* control the driver's interface to the Ultrastor 14F/34F series controllers. If you've installed multiple controllers, you can repeat these options for each controller. Remember, however, to specify the **us14scsi** keyword before each controller's set of options.

-a *address*	Set hex address of the controller to *address*, to be specified as a hexadecimal number.
-d *channel*	Set the DMA channel for bus mastering to *channel*.
-i *irq*	The interrupt used by the controller is *irq*.
-r	Reset the SCSI bus on driver startup.

-u *num* Set the maximum number of SCSI units to *num*. (Default 4)

Description:

Fsys.us14scsi is a SCSI driver for the Ultrastor US-14F and Ultrastor US-34F Host Bus Adapters.

The US 34F controller is a VESA local bus version of the US 14F.

For each controller, the driver performs a scan, looking for installed units. All targets are scanned (0 to 7) and for each target, each *LUN* (Logical Unit Number) is scanned (0 to 7). Devices are numbered starting from 0, and each type of device is numbered separately. For example, if you have a hard disk and a CD-ROM device, they'll be numbered **hd0** and **cd0**, respectively.

The Filesystem Manager supports only the following types of SCSI units:

0 **direct-access** (e.g. hard disk)

1 **sequential-access** (e.g. tape)

4 **WORM** (write once/read multiple)

5 **CD-ROM**

7 **optical**

Tape units

If a tape unit is found during the driver's scan or if you specify a sequential-access unit with the **-a** general option, two block special files are created under **/dev**. By default, they're named **/dev/tp0** and **/dev/tpr0**. Both these files reference the same physical drive, but **/dev/tpr0** will perform a rewind on a close.

CAUTION: To boot from a unit connected to the controller, make sure the controller is configured for a 64-head BIOS entry or specify the driver's **fsys -h** option.

Examples:

Support a US14 SCSI controller, list all connected devices:

```
Fsys.us14scsi &
```

Support a US14 SCSI controller, list all connected devices, and register direct access devices as **/dev/hard***n* and sequential access devices as **/dev/tape***n*:

```
Fsys.us14scsi fsys -n0=hard -n1=tape &
```

Support a US14 SCSI controller with 4 scsi units, don't display list of connected devices. Register direct access devices (e.g. hard disks) as **/dev/scsi***n*:

```
Fsys.us14scsi -q fsys -n0=scsi us14scsi -u4 &
```

Support US14 SCSI controllers, scan I/O ports which are available for use to find all existing controllers. Reset controllers at initialization:

```
Fsys.us14scsi us14 -s -r &
```

Files:

Fsys.us14scsi closes its standard input, standard output and standard error immediately after completing its initializations. Error messages may be produced during the initialization phase and will be written to standard error.

Fsys.us14scsi will cause **Fsys** to adopt various block special devices under **/dev**. These devices will normally be named **hd***n*, where *n* is the physical unit number of the device. The base portion of this filename (**hd**) may be overridden by the **-n** option.

Exit status:

Fsys.us14scsi will terminate only if an error occurs during startup, or if it has successfully forked itself upon startup because it had not been initially started in the background.

0 **Fsys.us14scsi** was not started in the background and as a result forked itself. The original process terminated with a zero exit status, the forked process continued.

>0 An error occurred during startup.

Caveats:

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

Fsys.vpm50

© 2005, QNX Software Systems

Driver for IBM Valuepoint ATA fixed-disk interface (QNX)

Syntax:

Fsys.vpm50 [*options***...]**

Options:

-a *controller*	Assume the controller is at this address (default is 1f0 hex).
-B	Ignore BIOS settings for the drive. (Default with **Fsys.vpm50**)
-c *drivespec*	Assume these parameters for the drive. You specify these in the following format: *heads*,*cylinders*,*sectors*[,*disknum*] You must include *heads*, *cylinders*, and *sectors*. If you omit *disknum*, unit 0 is assumed. (Default: autodetect)
-f *fsysname*	Registered name of filesystem process to attach to. (**qnx/fsys32**)
-i *irq*	Assume the controller is using this interrupt (default is 14).
-L	Do not use LBA.
-m *maxsize*	Allow no more than this many blocks to be transferred from the disk to memory in one operation (default is 127).
-M *multi_limit*	Maximum number of blocks per interrupt. (Default: maximum support by the drive.)
-n *device_name*	Assign this name to the block special device associated with the hard disk. The default is **hd**, which yields a device name of **/dev/hd***n*, where *n* is a unit number assigned by **Fsys**.

-N *version_name*
: Assign this unique version name to this invocation of the driver. The name can be from 1 to 11 characters in length. (Default: IDE)

-p
: Program the drive to use the highest I/O mode it is capable of.

-R *nretries*
: Number of times to retry bad blocks (default 1)

-v
: Verbose — report information about drives on startup.

-V
: Do not scan for CMD-640 VL interface.

Description:

Fsys.vpm50 is the driver for the AT-Attachment (ATA) disk drive interface on IBM Valuepoint machines. The ATA technology integrates the traditionally host-based controller electronics within the drive itself.

In order to obtain drive parameters, **Fsys.vpm50** examines values established by the BIOS of your computer. If necessary, you can override these default parameters with the **-c** option.

Fsys.vpm50 will use LBA if the drive supports it. If the **-p** option is specified, the driver will use the highest PIO mode supported by the drive.

LBA is required if the drive is greater than 528M and DOS has been installed on the drive.

To define the largest single read or write the driver will support, use the **-m** *maxsize* option; the default value for this option is 127.

To help prevent you from unintentionally starting the same driver twice, **Fsys** rejects the invocation of any driver that has the same version name as a previously mounted driver. So if you wish to run multiple copies of **Fsys.vpm50**, you must use the **-N** *version_name*

option to assign a unique version name to the driver. (You can display the version name of a driver with the **sin ver** command.) You should also use the **-n** *device_name* option to specify a unique device name, since another driver may have already created the default device name.

Caveats:

Some IDE drives don't match the draft standard that **Fsys.vpm50** assumes (*AT Attachment Interface for Disk Drives*, Rev 4, March 17, 1993). If a drive won't function with **Fsys.vpm50**, please inform our technical support group. We'll need the make, model, and year of purchase of your drive.

Don't use **slay** to shut down a filesystem driver! If you do, new drivers may not be able to load properly and **Fsys** itself may experience internal errors.

If you need to shut down a filesystem driver, follow the steps in the "Driver shutdown" section in the documentation for **Fsys**.

See also:

Blkfsys, **Fsys**, **Fsys.***

fsysinfo

Display filesystem statistics (QNX)

Syntax:

fsysinfo [-b] [-d *secs***] [-r] [***file***]**

Options:

-b	Base the counts from the time **fsysinfo** starts (useful only with **-d**)
-d *secs*	Update the fullscreen display every *secs* seconds. To exit the display, you press Ctrl – C.
-r	Reset **Fsys**'s internal counters. To use this option, you must be logged in as **root**.
file	Display statistics for the filesystem containing this file (default is the filesystem which contains the current working directory).

Description:

The **fsysinfo** utility displays statistical information obtained from the Filesystem Manager (**Fsys**). The utility displays this information in three sections:

- configurable limits
- cache-and-disk statistics
- critical-function statistics

The **fsysinfo** utility displays this information only once unless you specify **-d**.

Because **Fsys** doesn't proportion its cache according to partitions, disks, or drivers, the statistics you see represent the total usage of the **Fsys** responsible for the named path (i.e. the statistics don't reflect usage for just that path).

If you specify the **-b** option, the statistics shown in the cache-and-disk and critical-function sections show activity from the time **fsysinfo** started. Otherwise, they show activity from when **Fsys** started or from when you reset **Fsys**'s counters with **fsysinfo -r**.

Configurable limits

The configurable limits section shows information for the following configurable **Fsys** resource limits:

Resource	Fsys option for configuring limit
inodes	**-i**
names	**-C**
file descriptions (open control blocks)	**-f**
locks	**-l** ("el")
threads	**-t**

For each of these configurable limits, the display shows the:

- current value
- greatest value that has been used at one time
- limit set by **Fsys**

If your machine is tight on memory you can use this information to tune your **Fsys** limits. You should find the default values generous for most systems.

Cache-and-disk statistics

The cache-and-disk statistics can help you tune the cache size to obtain the best performance. Generally, the higher you raise the cache hit ratio (displayed as a percentage on the first line of these statistics), the better the performance.

This statistic:	Represents the number of:
data hits	times **Fsys** had to read a block and found that block already in cache. A cache hit may occur when a program does a read or readdir.
data delays	times a driver started reading a requested data block that wasn't yet available (**Fsys** performs a pre-read when it recognizes that the file usage pattern is sequential). If **Fsys** pre-reads a single block, any subsequent blocks for the read request that reside in cache are also considered delayed and are added to this number.
data misses	times the requested data block wasn't in cache, and **Fsys** had to start a disk read.
metadata hits	times **Fsys** had to read a block in the process of obtaining metadata (data about the file required to open, stat, get extent information and so on) and found that block already in cache.
metadata misses	times the requested block block of metadata wasn't in cache, and **Fsys** had to start a disk read.
cache lookups	reads, writes, and other internal functions that have been performed by **Fsys** (does *not* represent the sum of hits, delays, and misses).
cache writes	cache blocks that have been queued for writing to disk. Because of write buffering, a block may be queued many times before hitting the disk.
disk reads	block read requests **Fsys** has made to its drivers.

continued...

This statistic:	Represents the number of:
disk writes	block write requests **Fsys** has made to its drivers.

Critical-function statistics

These statistics represent how often certain functions in **Fsys** have been called. You can't use this information for tuning purposes, but it does give an idea of how busy the system is.

With the exception of *namei()*, which is the internal function used to map a pathname to an in-memory inode, all the critical function names have a one-to-one correspondence with library functions of the same name.

When you run **fsysinfo** in fullscreen display mode (**-d**), the critical-functions section displays:

- the number of times each function was called
- a per-second count of the number of calls since the last display
- the average number of calls per second since the **Fsys** counters were initialized (or since **fsysinfo** was started if started with **-b**)

When you don't run **fsysinfo** in fullscreen mode, it displays only the number of times each function was called and the average number of calls per second.

Examples:

Display statistics for the current directory:

```
fsysinfo
```

Reset the stat counters in the **Fsys** process that's responsible for **/tmp**:

```
fsysinfo -r /tmp
```

Update the statistical display every three seconds, showing **Fsys** statistics for **/**:

```
fsysinfo -d3 /
```

See also:

Fsys

fullpath

© 2005, QNX Software Systems

Display network-qualified pathnames (QNX)

Syntax:

fullpath [-t] [*name***...]**

Options:

-t Write terse output only, i.e. write only the fully qualified path; default is to write:

name **is** *qualified_path***\n**

name A filename.

Description:

The **fullpath** utility resolves all prefixes and symbolic links to display a fully qualified network path for each filename you specify. If no names are given, **fullpath** displays the full pathname of the current working directory.

Examples:

Get the full pathname of **/usr/lib/terminfo**:

```
$ fullpath /usr/lib/terminfo
/usr/lib/terminfo is //18/etc/terminfo
$
```

Get the full pathname of the current directory (which is for the purpose of this example the directory **/home/glen**):

```
$ cd /home/glen
$ fullpath
/home/glen is //61/home/glen
$
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

basename, **dirname**

gmake

© 2005, QNX Software Systems

Maintain, update, and regenerate groups of programs (GNU)

Syntax:

make [-einpqrst] [-f *makefile*]... **[-k | -S]**
[*macro*=*name*...] [*target_name*...]

Options:

See **make**.

Description:

The QNX **make** utility is a port of GNU's **gmake**, which has an extensive set of additional features beyond what POSIX defines. The full documentation for GNU **make** is available from the Free Software Foundation.

Author:

GNU

See also:

make

Andrew Oram and Steve Talbott, *Managing Projects with make*, O'Reilly and Associates, 1991.

Syntax:

grep [-E|-F] [-c|-f|-q] [-hinsvx]
[-e *expression***|-f** *expression_file***]... [***file***...]**

grep [-E|-F] [-c|-f|-q] [-hinsvx]
expression **[***file***...]**

Historical UNIX versions:

egrep [-chilnqsvx] [-e *expression***|-f** *expression_file***]...**
[*file***...]**

egrep [-chilnqsvx] *expression* **[***file***...]**

fgrep [-chilnqsvx] [-e *expression***|-f** *expression_file***]...**
[*file***...]**

fgrep [-chilnqsvx] *expression* **[***file***...]**

Options:

-c Write only a count of selected lines to standard output.

-e *expression* A regular expression, of type determined by the **-E**, **-F** options. This form is used when more than one expression needs to be specified on the command line. The **-e** option can be repeated.

-E Use extended regular expression (ERE) syntax.

-f *expression_file*

File containing a set of regular expressions, each separated by a **newline**. The type of the expressions is determined by the **-E** and **-F** options. This form is used when more than one

expression needs to be specified. The **-f** option can be repeated.

-F Treat *expression* as a fixed string instead of a regular expression. (Search for a fixed string or fixed strings.)

-h (QNX extension) Don't prefix matched lines with a filename. This option applies only when **grep** is invoked with more than one file to search.

-i Ignore uppercase and lowercase distinctions during comparisons.

-l ("el") Write only the names of files containing selected lines to standard output.

-n Before each output line, display the line's line number in the file.

-q Be quiet; don't write anything to the standard output, regardless of matching lines.

-s Suppress the error messages ordinarily written for non-existent or unreadable files. Other error messages aren't suppressed.

-v Select only those lines that don't match the specified patterns.

-x Consider as matching lines only input lines selected against an entire fixed string or regular expression.

expression A regular expression, whose type is determined by the **-E** and **-F** options. This form is used when only one expression is specified on the command line. Any names specified after this option are treated as input files.

file The text file to be input. The default is standard input.

Description:

The **grep** utility searches input for lines matching the expression(s) given. When an input line matches any of the expressions, it is said to be "selected." By default, selected lines are written to standard output.

Numerous options allow variations upon the output format. For example, to reverse the meaning of the output, the **-v** option could be used.

There are three types of expressions understood by **grep**: Basic Regular Expressions, Extended Regular Expressions, and Fixed Regular Expressions. If you don't specify **-E** or **-F**, the expression(s) are taken to be Basic Regular Expressions.

Basic and Extended Regular expressions are similar to arithmetic expressions in that larger expressions are formed by combining smaller expressions and operators according to some precedence rule.

Regular expressions have an "invisible" operator, i.e. concatenation. The concatenation of two expressions means match the one on the left, then the one on the right.

The smallest expression is a single character.

Basic regular expressions

The following table summarizes the Basic Regular Expressions, and the precedence of the operators:

Expression	Meaning
\(*expression* **\)**	subexpression. Match the pattern *expression*. Used for backreferences (see below), and precedence
N	back-reference. Match the exact string that the *N*th subexpression did

continued...

Expression	Meaning
.	(dot) match any single character
[*charset***]**	match any member of the set *charset* (see below)
c	match any non-special character
c	Match literal *c*. The character may not be **(**, **)**, **{**, **}**, or any digit from **1** through **9**. The \ is usually used to escape *****, **$**, **^**, **.**, **[** and **]**. \\ matches a literal "\". \ has no special meaning inside a bracket expression.
*limited_expression******	match any number of repetitions of *limited_expression* including zero.
*limited_expression***\{***M***\}**	match exactly *M* repetitions of *limited_expression*
*limited_expression***\{,***N***\}**	match zero to *N* repetitions of *limited_expression*
*limited_expression***\{***M***,***N***\}**	match *M* to *N* repetitions of *limited_expression*
expr0expr1	(concatenation) match *expr0* then *expr1*
^*expression*	match *expression* only at beginning of line
*expression***$**	match *expression* only at end of line

A *limited_expression* is restricted to a back-reference, a subexpression, or a BRE matching a single character.

A *charset* is formed by concatenation of the following operators:

Expression	Meaning
c	any character *c*
c-*d*	any character in the range from *c* to *d*
[:alpha:]	any alphabetic character
[:upper:]	any uppercase character
[:lower:]	any lowercase character
[:digit:]	any numeric character
[:alnum:]	any numeric or alphabetic character
[:xdigit:]	any character used to represent a hexadecimal number
[:space:]	any character that is a whitespace
[:print:]	any printable character
[:punct:]	any character that is punctuation
[:graph:]	any character with a graphic representation
[:cntrl:]	any character used for control

If the *charset* begins with the caret (^), the set is inverted. For example:

```
[^[:alpha:]]
```

means match any non-alphabetic character. (This can also be expressed by **[^a-zA-Z]**.)

Extended Regular Expressions

The Extended Regular Expressions are an enriched set of regular expression operators. In particular, the Extended Regular Expressions support an operator for alternation, thus allowing a match of one expression or another. It is also important to note that the parenthesis syntax is different from Basic Regular Expressions, and the semantics

are subtly different. There are no back-references in Extended Regular Expressions.

The following list summarizes the Extended Regular Expressions:

Expression	Meaning
(*expression***)**	match *expression*; useful for altering precedence
.	(dot) match any single character
c	match any non-special character *c*
c	Match literal *c*. Normally used to escape ERE special characters.
[*charset***]**	match any element of *charset*
*limited_expression******	match any number of repetitions of *limited_expression*, including zero
*limited_expression***+**	match 1 to any number of repetitions of *limited_expression*
*limited_expression***?**	*limited_expression* is optional (match 0 or 1 repetition)
*limited_expression***\{***M***\}**	match exactly *M* repetitions of *limited_expression*
*limited_expression***\{,***N***\}**	match zero to *N* repetitions of *limited_expression*
*limited_expression***\{***M***,***N***\}**	match *M* to *N* repetitions of *limited_expression*
expr0expr1	(concatenation) match *expr0* then *expr1*
*expr0***|***expr1*	(alternation) match *expr0 or expr1* (not both)

continued...

Expression	Meaning
^*expression*	match *expression* only at the beginning of a line
expression$	match *expression* only at the end of a line

For extended regular expressions a *limited_expression* is restricted to an expression matching a single character or an expression enclosed in parentheses.

Fixed Regular Expressions

Fixed Regular Expressions consist of a set of strings of characters. They do not permit the operators of Extended or Basic Regular Expressions. The algorithm used is extremely efficient for locating one of a set of strings within another string. Thus, if you don't need the various operators of Basic or Extended Regular Expressions, the Fixed Expressions are a better choice.

Examples:

Display lines in **Phone.List** containing telephone numbers:

```
grep '[[:digit:]]\{3\}-[[:digit:]]\{4\}' Phone.List
```

Display all the people logged in who are listed in the **MyFriends** file:

```
who | grep -F -f MyFriends
```

Display all occurrences of the words "steve" and "barney" in the **Phone.List** file:

```
grep -F -e steve -e barney Phone.List
```

Files:

The **grep** utility will read lines from the file(s) named on the command line including the standard input for each command-line item which is **-**, or from the standard input if no files are named on the command line.

Lines matching the search criteria will be written to the standard output. When multiple input files have been specified on the command line, each line in the output will be preceded by the name of the source file and a colon (**:**). If the input file was standard input specified by a dash (**-**) on the command line, the line will be prefixed by "**stdin:** ".

If an error occurs a diagnostic message is written to the standard error.

Exit status:

0 Lines were found matching the expression provided.

>0 An error occurred or no matching lines were found.

See also:

awk, **lex**, **sed**

gunzip

Uncompress files (GNU)

Syntax:

gunzip [-cfhLrtvVM] [*name***...]**

Options:

See **gzip/gunzip/zcat**.

Description:

The **gunzip** utility uncompresses files that have been compressed with **gzip**. It is equivalent to **gzip -d**. See **gzip** for more details.

Author:

GNU

See also:

gzip, **freeze**, **pax**

gzip/gunzip/zcat

© 2005, QNX Software Systems

Compress or expand files (GNU)

Syntax:

Compress/uncompress files:

gzip [-cdfhLrtvV19] [*name***...]**

Uncompress only:

gunzip [-cfhLrtvVM] [*name***...]**

Catenate compressed files:

zcat [-hLV] [*name***...]**

Options:

-c Write output on standard output; keep original files unchanged. If there are several input files, the output consists of a sequence of independently compressed members. To obtain better compression, concatenate all input files before compressing them.

-d Decompress.

-f Force compression or decompression even if the file has multiple links or the corresponding file already exists. If **-f** is not given, and when not running in the background, **gzip** prompts to verify whether an existing file should be overwritten.

-h Display a help message.

-L Display the **gzip** license.

-q Suppress all warnings.

-r Travel the directory structure recursively. If any of the file names specified on the command line are directories, **gzip** will descend into the directory and compress all the files it finds there (or decompress them in the case of **gunzip**).

-t Test. Check the compressed file integrity.

-v Verbose. Display the name and percentage reduction for each file compressed.

-V Version. Display the version number and compilation options.

-*n*
-fast
-best Regulate the speed of compression using the specified digit *n*, where **-1** (the number one) or **-fast** indicates the fastest compression method (less compression) and **-9** or **-best** indicates the slowest compression method (optimal compression). The default compression level is **-5**...

Description:

The **gzip** utility reduces the size of the named files using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension **.gz**, while keeping the same ownership modes, access and modification times. If no files are specified, the standard input is compressed to the standard output. If the new file name is too long, **gzip** truncates it and keeps the original file name in the compressed file. The **gzip** utility will attempt to compress only regular files. In particular, it will ignore symbolic links.

Compressed files can be restored to their original form using **gzip -d** or **gunzip** or **zcat**.

The **gunzip** utility takes a list of files on its command line and replaces each file whose name ends with **.gz** or **.GZ** or **-z** and which begins with the correct magic number with an uncompressed file without the original extension. **gunzip** also recognizes the special extensions **.tgz** and **.taz** as shorthands for **.tar.gz** or **.tar.GZ**.

The **gunzip** utility can currently decompress files created by **gzip**, **zip**, **compress** or **pack**. The detection of the input format is automatic. When using the first two formats, **gunzip** checks a 32 bit CRC. For **pack**, **gunzip** checks the uncompressed length. The

compress format was not designed to allow consistency checks. However **gunzip** is sometimes able to detect a bad **.GZ** file. If you get an error when uncompressing a **.GZ** file, do not assume that the **.GZ** file is correct simply because the standard **uncompress** does not complain. This generally means that the standard **uncompress** does not check its input, and happily generates garbage output.

Files created by **zip** can be uncompressed by **gzip** only if they have a single member compressed with the "deflation" method. This feature is intended only to help conversion of **tar.zip** files to the **tar.gz** format. To extract **zip** files with several members, obtain and use **unzip** instead of **gunzip**. (Note that **unzip** is not shipped as part of QNX.)

The **zcat** utility is identical to **gunzip -c**. The **zcat** utility uncompresses either a list of files on the command line or its standard input and writes the uncompressed data on standard output. **zcat** will uncompress files that have the correct magic number whether they have a **.gz** suffix or not.

The **gzip** utility uses the Lempel-Ziv algorithm used in **zip** and **PKZIP**. The amount of compression obtained depends on the size of the input and the distribution of common substrings. Typically, text such as source code or English is reduced by 60-70%. Compression is generally much better than that achieved by LZW (as used in **compress**), Huffman coding (as used in **pack**), or adaptive Huffman coding (**compact**).

Compression is always performed, even if the compressed file is slightly larger than the original. The worst case expansion is a few bytes for the **gzip** file header, plus 5 bytes for every 32K block, or an expansion ratio of 0.015% for large files. The **gzip** utility preserves the mode, ownership and timestamps of files when compressing or decompressing.

Multiple compressed files can be concatenated. In this case, **gunzip** will extract all members at once. For example:

```
gzip -c file1  > foo.gz
gzip -c file2 >> foo.gz
```

Then:

```
gunzip -c foo
```

is equivalent to:

```
cat file1 file2
```

In case of damage to one member of a **.gz** file, other members can still be recovered (if the damaged member is removed). However, you can get better compression by compressing all members at once:

```
cat file1 file2 | gzip > foo.gz
```

compresses better than:

```
gzip -c file1 file2 > foo.gz
```

If you want to recompress concatenated files to get better compression, do:

```
zcat old.gz | gzip > new.gz
```

Environment variables:

The environment variable **GZIP** can hold a set of default options for **gzip**. These options are interpreted first and can be overridden by explicit command line parameters. For example:

```
GZIP="-8 -v"; export GZIP
```

© 2005, QNX Software Systems

Exit status:

2 The operation succeeded but perhaps not 100%; a warning was generated in the process.

1 An error occurred; the operation failed.

0 The operation succeeded.

Author:

GNU

Caveats:

The **.gz** extension is already also used by UNIX **pack**. You can link **gzip** to **pcat** to get transparent decompression for programs expecting **.gz** files to be in pack format.

See also:

pax, **freeze**

Syntax:

hd [-A *format***] [-n** *count***] [-s** *skip***]**
[-t *fmt_string***] [-v] [**_file_**...]**

Options:

-A *format* Display the file offset field in the specified format. Valid formats are:

format	Display
x	(hex, 7 digits)
o	(octal, 10 digits)
d	(decimal, 9 digits)
n	(none)

-n *count* Display only the first *count* bytes (default), blocks kilobytes or megabytes, depending on the trailing character (**b** for blocks, **k** for kilobytes, **m** for megabytes).

-s *skip* Ignore the first *skip* bytes (default), blocks, kilobytes or megabytes, depending on the trailing character (**b** for blocks, **k** for kilobytes, **m** for megabytes).

-t *fmt_string* Use this output/display format.

-v Be verbose. Display all input.

file The pathname of an input file. If no files are specified, the standard input is used. If a *file* is a dash character (**-**), **hd** reads from the standard input at that point in the sequence.

© 2005, QNX Software Systems

Description:

You use the **hd** utility to output data in decimal, hex, octal, or ASCII. The name "**hd**" (hex dump) is derived from the default output format.

The **hd** utility processes input in 16-byte units that are formatted into a line. In the default output format:

- the file offset field is displayed in hex, 7 digits
- a colon and a space separates the file offset field from the data
- the data is displayed as 16 space-separated bytes in hex
- the same data is also displayed in ASCII, if printable; unprintable data appears as dots

For example, if you issue the following command:

```
echo "abcdefghijklmnopqrstuvwxyz01234" | hd
```

you should see the following output:

```
0000000: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 abcdefghijklmnop
0000010: 71 72 73 74 75 76 77 78 79 7a 30 31 32 33 34 0a qrstuvwxyz01234.
```

The **-v** option (verbose) inhibits **hd**'s default behavior of folding multiple identical lines into a single line that contains an asterisk (*****). When **-v** is specified, all data is displayed.

To exclude part of the input, you use the **-n** and **-s** options. You can specify the arguments to these options in hex (using a **0x** prefix) or octal (using a **0** prefix).

The arguments to **-n** and **-s** also let you specify three different units:

To specify:	Add this suffix:
blocks (512 bytes)	**b**
kilobytes (1024 bytes)	**k**
megabytes (1048576 bytes)	**m**

To specify the output format, you use the **-t** option. The *fmt_string* argument — which can be specified in decimal, hex, or octal — instructs **hd** as to which format to use for presenting the output. The *fmt_string* argument can be one of the following:

***fmt_string*:**	**Display as:**
o[**1**|**2**|**4**]	octal, 1-, 2-, or 4-byte objects (default is 2)
x[**1**|**2**|**4**]	hex 1-, 2-, or 4-byte objects (default is 2)
X[**1**|**2**|**4**]	hex 1-, 2-, or 4-bytes objects, with characters (default is 1)
d[**1**|**2**|**4**]	decimal 1-, 2-, or 4-byte objects (default is 2)
c	characters

The input, processed in 16-byte units formatted into a line, is displayed according to the option ([**1**\|**2**\|**4**]) you choose:

To display input as:	**Choose:**
sixteen 1-byte objects	**1**
eight 2-byte objects	**2**
four 4-byte values per line	**4**

The **X** (uppercase) format instructs **hd** to redisplay the input block at the end of line in a special character format. The special character format displays printable characters “as is” and nonprintable characters as a single dot (**.**). If you don’t specify the **-t** option, the default output format is the same as using the option **-t X**.

The **c** format displays printable characters as themselves. With the exception of the following characters, all other characters are displayed as 2-digit hex values.

© 2005, QNX Software Systems

ASCII mnemonic:	Value:	Representation:
NUL	00	\0
alert	07	\a
backspace	08	\b
tab	09	\t
newline	0a	\n
vertical tab	0b	\v
formfeed	0c	\f
carriage return	0d	\r

Examples:

Display the 2nd to 11th sectors of the floppy disk **/dev/fd0**:

```
hd -s 1b -n 10b /dev/fd0 | more
```

Exit status:

0 All input files were processed successfully.

>0 An error occurred.

See also:

cat, **head**, **tail**, **spatch**

head

Copy the first part of files (POSIX)

Syntax:

head [-c|-l] [-n *number***] [***file***]...**

Options:

-c	Measure the quantity of output in bytes.
-l	("el") Measure the quantity of output in lines; this is the default unit of measure.
-n *number*	The number of lines (default) or bytes output. The *number* argument is an unsigned decimal integer. If you omit the **-n** *number* option, the default is the first ten units (lines or bytes) of the file.
file	The pathname of an input file. If you don't specify any files, the standard input is used.

Description:

The **head** utility copies its input files to the standard output. The utility ends the output for each file at a point designated by the **-n** *number* option. The *number* argument is counted in bytes if you choose the **-c** option, and in lines if you choose the **-l** option.

If you specify multiple files, **head** prints an identifying header before the output for each file.

Examples:

Display the first ten lines of all files in the current directory:

```
head *
```

Print the first 16 bytes of **myfile** in hex:

```
head -c -n 16 myfile | hd
```

(Note that in this case, the same functionality is offered through command-line options to **hd**.)

© 2005, QNX Software Systems

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

tail

Syntax:

id [*username***]**

id -G [-n] [*username***]**

id -g [-nr] [*username***]**

id -u [-nr] [*username***]**

Options:

-g Write the effective group ID only.

-G Write the real (and effective, if different) group ID only.

-u Write the effective user ID only.

-n In combination with **-G**, **-g**, or **-u**, write the ID as its name instead of as an unsigned integer.

-r In combination with **-g** or **-u**, write the real ID instead of the effective ID.

Description:

The **id** utility writes the current real and/or effective user ID and group ID.

When no options are specified, output is as follows:

uid=*nnn***(***username***) gid=***nnn***(***groupname***)**

If the effective user ID is different from the real user ID, the effective user ID will also appear:

... **euid=***nnn***(***effective_username***)**

Likewise, if the effective group ID is different from the real group ID, the effective group ID will appear as well:

© 2005, QNX Software Systems

... **egid=***nnn***(***effective_groupname***)**

If no entry exists for an ID in the **/etc/passwd** or **/etc/group** files, the output will lack the **(***name***)** after the numerical value. No error is generated.

If a *username* is supplied as an operand, the effective *uid* and *gid* won't appear since there's no process associated with it — the information is simply looked up in the **/etc/passwd** and **/etc/group** files.

When options are specified, the requested data is written as an unsigned integer, unless option **-n** is specified, in which case the data is written as the corresponding user or group name. With option **-G**, the **id** utility might produce two lines of output (one value per line) if the real and effective IDs differ. In all other cases, **id** always produces one line of output.

Examples:

Write information on current IDs (real and effective):

```
$ id
uid=109(eric) gid=120(techies)
```

Write the effective group ID as a number:

```
$ id -g
120
```

Write the effective group ID as a name:

```
$ id -gn
techies
```

Files:

/etc/passwd	Password file; defines user IDs, home directories, etc.
/etc/group	Group file; defines the valid group IDs for the system, also lists user IDs who may change to each group.

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

logname, **who**

ident

© 2005, QNX Software Systems

Identify files (UNIX)

Syntax:

Display **ident**'s version number:

```
ident -V
```

Identify files:

```
ident [-q] [file]...
```

Options:

-q Be quiet; suppress warning message if no patterns are found.

-V Display **ident**'s version number.

file The pathname of a file to be searched. If no files are specified, **ident** uses the standard input.

Description:

The **ident** utility searches for all occurrences of the pattern **$***keyword***:...$** in each named file. These patterns are normally inserted automatically by the RCS command **co**, but can also be inserted manually.

The **ident** utility works on text files as well as object files and dumps. For example, if the C program in **f.c** contains

```
char rcsid[] = "$Id: f.c,v 5.0 1997/08/22 09:09:36
eggert Exp $";
```

and **f.c** is compiled into **f.o**, then the command:

```
ident f.c f.o
```

will write the following output:

```
f.c:
    $Id: f.c,v 5.0 1997/08/22 09:09:36 eggert Exp $
f.o:
    $Id: f.c,v 5.0 1997/08/22 09:09:36 eggert Exp $
```

Author:

GNU

See also:

ci, **co**, **rcs**, **rcsdiff**, **rcsmerge**, **rlog**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience 15, 7*, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly and Associates, 1995.

infocmp

© 2005, QNX Software Systems

Output the contents of a terminfo capability file (UNIX)

Syntax:

infocmp [-1CILQ] [-w *width***] [***terminal_name***]**

Options:

-1	("one") Force the output to contain one entry per line. Without this option, output will be printed to a maximum width of 60 characters, each line containing as many entries as can fit on that line.
-C	Generate output in **termcap** format.
-I	Generate output in **terminfo** format. This option is the default.
-L	Use the long C variable names listed in **/usr/include/term.h**
-Q	Use the names that map onto the QNX console capabilities.
-w *width*	Force the output to the specified number of columns. As many entries as will fit on each line will be output.
terminal_name	The name of a terminal capability file to display. If the **TERMINFO** environment variable is *not* set, the file will be read from the **/usr/lib/terminfo** directory in a subdirectory named by the first letter of the terminal name. For example, the VT100 **terminfo** file is stored in: **/usr/lib/terminfo/v/vt100**. If set, the **TERMINFO** environment variable will define a directory where the compiled description file is read from. If a *terminal_name* is not specified, the terminal named in the **TERM** environment variable will be used.

Description:

The **infocmp** utility is used to output the contents of a previously compiled **terminfo** capability file in a number of formats. The default format is suitable for editing and recompilation with the **tic** utility. The output is sorted such that the Boolean capabilities are output first, followed by the integer capabilities and then the string fields. The **infocmp** utility, executed without any options, will produce the **terminfo** description of the currently defined terminal in a form suitable for editing and recompilation with the **tic** utility.

If no options are specified and zero or one terminal name is specified, the **-I** option is assumed.

See also:

tic, **terminfo** technical note (available on QUICS)

John Strang, Linda Mui, & Tim O'Reilly, *termcap & terminfo*, O'Reilly and Associates, 1988.

Input

© 2005, QNX Software Systems

Start the Input Manager

Syntax:

Input [-bdgltv] *protocol* **[***protocol_opts***...]**
[*protocol* **[***protocol_opts***...]]...**
[*device* **[***device_opts***...]]... [***filter* **[***filter_opts***...]]...**

Options:

-b Prevent Ctrl – Alt – Shift – Backspace (CASB) exit (default: allow CASB).

-d Device (default: **/dev/photon** or **$PHOTON**).

-g Input group (default: 1).

-l List the internal modules. Modules are listed in the following format where **class** is one of: **D** - Device, **P** - Protocol, or **F** - Filter:

```
module name | date last compiled | revision | class
```

-t Throttle factor in milliseconds (default: 0).

-v Increment verbosity (default: 0).

***protocol* [*protocol_opts*...]**

ahl [**-b** *baud*] AHL-51A touchscreen protocol.

Options:

-b *baud* Baud rate (default: 9600).

Devices supported: **fd**, **uart**

airmse [**-b** *baud*]

AirMouse protocol.

Options:

-b *baud* Baud rate (default: 1200).

Devices supported: **fd**, **uart**

busmse [**-i** *irq*] [**-p** *ioport*]

Microsoft bus mouse controller.

Options:

-i *irq* IRQ for bus adapter (default: 5).

-p *ioport* Port of bus adapter (default: 0x23c).

Devices supported: **busmse** controller only.

carrol [**-b** *baud*]

Carroll Touch SmartFrame protocol.

Options:

-b *baud* Baud rate (default: 2400).

Devices supported: **fd**, **hbc**, **uart**

compaq Compaq Concerto protocol.

Devices supported: **spratly**

cyber [**-[A-Z]** *btnmap*] [**-b** *baud*]

CyberScan 3232 protocol.

Options:

-[A-Z] *btnmap* Define virtual button mapping (e.g. **ulx,uly:lrx,lry:btn**).

-b *baud* Baud rate (default: 9600).

Devices supported: **fd**, **uart**

devmse [**-d** *name*]

Input provided by a resource manager.

Options:

© 2005, QNX Software Systems

-d *name* Device name provided by resource manager (no default).

Devices supported: **devmse** controller only.

dyna [**-b** *baud*] Dynapro SC3 protocol.

Options:

-b *baud* Baud rate (default: 2400).

Devices supported: **fd**, **uart**

gm [**-b** *baud*] GM Nameplate protocol.

Options:

-b *baud* Baud rate (default: 9600).

Devices supported: **fd**, **uart**

intellitch [**-b** *baud*] [**-h** *press*] [**-r** *milliseconds*] [**-u**] [**-z**]
Elographics IntelliTouch or AccuTouch protocols.

Options:

-b *baud*	Baud rate — if connected to a serial input.
-h *press*	Simulate a right button when the pressure is greater than *press*.
-r *milliseconds*	Number of milliseconds that finger must be released for an untouch (left button) event to occur (default: 80).
-u	Disable untouch (left button) event generation.
-z	Enable z “pressure” data — only on 40XX controllers.

Devices supported: **elo -I**, **fd**, **uart**

kbd [**-k** *rate*[,*delay*]] [**-R**]

Keyboard scan codes (connected to primary keyboard).

Options:

-k *rate*[,*delay*]	Keyboard rate (Hz)/delay(ms) (default: 40Hz, 1000ms).
-R	Don't reset keyboard (default: reset keyboard).

Devices supported: **kb**

microtch [**-b** *baud*]

Microtouch protocol.

Options:

-b *baud*	Baud rate (default: 9600).

Devices supported: **fd**, **uart**

msoft [**-3**] [**-b** *baud*] [**-i**][**-R**]

Microsoft-compatible mouse protocol (serial).

Options:

-3	Microsoft 3-button mouse.
-b *baud*	Baud rate for serial device (default: 1200).
-i	Intellimouse.
-R	Don't reset mouse (default: reset mouse).

Devices supported: **fd**, **uart**

msys [**-b** *baud*] Mouse Systems mouse protocol (used by Logitech).

Options:

© 2005, QNX Software Systems

-b *baud* Baud rate for serial device (default: 1200).

Devices supported: **fd**, **uart**

ps2 [**-r**] IBM PS/2 mouse protocol.

Options:

-r Reset the mouse (default: don't reset mouse).

Devices supported: **busmse**, **kb -2**

smartset [**-b** *baud*]

Elographics Smartset protocol.

Options:

-b *baud* Baud rate (9600).

Devices supported: **elo**, **fd**, **uart**

wacom [**-b** *baud*] [**-P**]

Wacom PL-100V protocol.

Options:

-b *baud* Baud rate (default: 19200).

-P Enable pressure mode (default: automatically determine).

Devices supported: **fd**, **uart**

All serial devices use 8 data bits, 1 stop bit, and no parity.

***device* [*device_opts*...]**

elo [**-I**] [**-i** *irq*] [**-p** *ioport*]

Elographics 14X and 40XX bus driver.

Options:

-I	40XX (IntelliTouch board).
-i *irq*	IRQ for bus adapter (default: 5).
-p *ioport*	Port of bus adapter (default: 0x280).

fd [**-d** *device*] Opens a device via *open()*.

Options:

-d *device*	Device to open **fd** on (default: **/dev/ser1**)

hbc [**-i** *irq*] [**-p** *ioport*]

Carroll Touch Hardware Based Controller.

Options:

-i *irq*	HBC interrupt (default: 7).
-p *ioport*	HBC port (default: 0x300).

kb [**-2**] [**-i** *irq*] [**-p** *ioport*]

Keyboard controller.

Options:

-2	Set default IRQ and I/O port for auxiliary port.
-i *irq*	IRQ (default: 1).
-p *ioport*	Port (default: 0x60).

spratly [**-i** *irq*] [**-p** *ioport*]

Compaq Spratly interface.

Options:

© 2005, QNX Software Systems

-i *irq* Spratly interrupt (default: 15).

-p *ioport* Spratly port (default: 0x2065).

uart [**-i** *irq*] [**-p** *ioport*]

Accesses 8250/16450/16550 UART directly.

Options:

-i *irq* IRQ for serial device (default: 4).

-p *ioport* Port of serial device (default: 3f8).

***filter* [*filter_opts*...]**

abs [**-b**] [**-c**] [**-f** *filename*] [**-o** *x,y*] [**-s** *x,y*]

Transform and compress absolute coordinate "touch" events.

Options:

-b Touching the screen emulates right mouse button (default: left).

-c Calibrate mode — don't transform coordinates.

-f *filename* Calibration filename.

-o *x,y* Origin of display region (default: the origin of the graphics region).

-s *x,y* Coordinates of lower-right corner of display region (default: the width and height of the graphics region).

keyboard [**-k** *kbd_file*]

Translate scan codes to Unicode.

Options:

-k *kbd_file* File to use for international keyboard mapping.

rel [**-G** *gain*] [**-l**] [**-T** *threshold*] [**-x**] [**-y**]

Filter and compress relative coordinate "mouse" events.

Options:

-G *gain*	Motion multiplier (default: 1).
-l	Swap right and left buttons.
-T *threshold*	Speed-doubling threshold in mickeys (100).
-x	Reverse X.
-y	Reverse Y.

Description:

The **Input** utility serves as an input driver for Photon. The **inputtrap** utility detects drivers and starts **Input**.

Input is responsible for taking data from an input device such as a mouse, touchscreen controller, or keyboard, interpreting the data, and then "doing" something with it. The default behavior is to package the data as an event and inject it into the Photon event space. But **Input** can also pass data to the **Mouse** and **Touch** I/O managers used in console, X, and QNX Windows applications.

Input uses a layered approach to driver design:

- filter layer
- protocol layer
- device layer

At each layer, data is interpreted/modified and passed up to the next layer until it's injected as an event into the Photon event space.

For each device **Input** talks to, there's a separate path through the three layers called an *event bus line*. An event bus line consists of modules, one representing each layer, linked together by a software bus. As data is passed up the layers via the bus, the data is manipulated by each module into a format recognizable by the next layer's module and so on.

It's important to note that you can run as many instances of **Input** as you want, one for each device. Or you can run one **Input** for all the devices. Choosing which scenario to use is mostly a matter of convenience.

For example, if you're trying to get an Elographics touchscreen to work, it's a good idea to run a separate **Input** for the touchscreen in order to get all the command-line arguments right. This way you won't hang your system if you put the wrong arguments in and **Input** dies — you'll still have your original **Input** talking to the mouse and keyboard.

Examples:

The **inputtrap** utility detects a Microsoft mouse interfaced directly through the UART, and a keyboard interfaced through the file descriptor provided by opening **/dev/kbd**. The **inputtrap** utility invokes **Input** with the following options:

```
Input kbd fd -d/dev/kbd msoft uart &
```

The **inputtrap** utility detects a PS/2 mouse interfaced through the auxiliary port on the keyboard controller (**kb -2**), and a keyboard interfaced through the primary keyboard port on the keyboard controller (**kb**). The **inputtrap** utility invokes **Input** with the following options:

```
Input kbd kb ps2 kb -2 &
```

Entering the following command causes **Input** to drive a Microsoft bus mouse, a keyboard (#1) interfaced through the primary keyboard port on the keyboard controller, and a keyboard (#2) connected through the auxiliary port on the keyboard controller:

```
Input kbd kb kbd kb -2 busmse &
```

See also:

inputtrap

inputtrap

Detect input devices and start Input Manager

Syntax:

```
inputtrap [-f device] [-k] [-l device] [-s] [-V...]
          [-v...] [-X class]... [-x device]...
          [start|query]
```

Options:

-f *device*	Probe for *device* first.
-k	Don't probe for a keyboard.
-l *device*	("el") Probe for *device* last.
-s	Safe mode (probe only for a keyboard).
-V	Pass a **-v** to **Input**.
-v	Increment verbosity level by 1.
-X *class*	Exclude any devices of *class* from scan.
-x *device*	Exclude *device* from scan.
query	Print command line necessary to start **Input**. If neither **start** nor **query** is specified, **query** is assumed.
start	Start driver.

Description:

The **inputtrap** utility probes for input devices and writes invocation commands along with the appropriate options and arguments for starting the found devices to the **/etc/config/trap/input.***node* file. The probing order is as follows:

1 Keyboard (**kbd**).

2 Devices owned by another manager; if found, terminate probing (**devmse**).

3 Inport bus mouse (**busmse**).

4 Microsoft bus mouse (**busmse**).

5 PS/2 mouse (**ps2**) .

6 UART-connected relative coordinate pointing devices (**uart**) .

7 **fd**-connected relative coordinate pointing devices (**fd**) .

8 UART- or **fd**-connected air mouse (**airmse**).

9 Elographics bus controller (**elo**).

The device names (given above) are recognized as arguments to **-f** and **-x**.

If a **/etc/config/trap/input.***node* file exists, **inputtrap** uses the contents of this file as arguments to **Input**.

Although Photon supports the following input hardware drivers, **inputtrap** doesn't detect all of the associated hardware devices:

Driver module	Detected?
ahl	no
airmse	yes
busmse	yes
carrol	bus interface adapters only
compaq	no
cyber	no
devmse	yes
dyna	no

continued...

Driver module	Detected?
gm	no
intellitch	bus interface adapters only
kbd	yes
microtch	no
msoft	yes
msys	yes
ps2	yes
smartset	bus interface adapters only
wacom	no

Examples:

Do a check for input devices. Once one of each class of device has been recognized, spawn **Input** with an appropriate command line:

```
inputtrap
```

The first device for each class of device recognized will be chosen (recognized classes are keyboard, relative, and absolute coordinate pointing devices). If a pointing device manager is running (i.e. **/dev/touch** or **/dev/mouse** exists), then assume this resource manager is properly configured — use its pointing devices and terminate probing.

The presence of **/dev/kbd** will not terminate probing.

Do a check (as above) for pointing devices. Don't start the driver, but print the command line required to start the driver:

```
inputtrap query
```

© 2005, QNX Software Systems

Do a check (as above) for input devices, starting with a check for a UART-based pointing device. This can be useful when you want a serial device to be detected instead of a builtin pointing device (e.g. on laptops and notebooks with builtin PS/2-style mice):

```
inputtrap -f uart start
```

Given the following **/etc/config/trap/input.***node* file:

```
kbd
fd -d/dev/kbd
intellitch -z
uart -2
```

running **inputtrap start** will start **Input** with an **fd**-based keyboard and an **intellitch** protocol, getting its input directly from a UART on COM port 2 (port 2f8 IRQ 3).

CAUTION:

Adding invalid information to the **input.***node* file can cause the system to hang. If your hardware changes, remember to remove the old **input.***node* file.

See also:

Input

/etc/install

Install a software package (QNX)

Syntax:

install [-u *update* **|** *drive***]** **[***files***...]**

Options:

-u *update*	Unpack and install the file *update*.
drive	Install a software package from the disk drive named by *drive*. Default is **/dev/fd0** (floppy disk drive 0).
files	Install only those files whose filenames match the pattern *files*. Default is ***** (install all files in the archive).

Description:

The **/etc/install** utility is a shell script provided as a convenience for installing QNX products and product updates on an already-running QNX system.

You must be **root** to run the **/etc/install** utility.

CAUTION: The **install** utility will typically install a directory tree from your current working directory. To install a product or update obtained from QNX Software Systems Ltd you will typically want to be in the root directory (**/**) before you run **/etc/install**.

It is still possible for an archive being installed to install components elsewhere than under the current working directory — check any installation instructions that came with the product or archive.

Disk updates are also unbundled using the **/etc/install** script. In this case, the updates will be read from the first floppy drive (**/dev/fd0**) or the *drive* named as a command-line parameter.

The **/etc/install** utility will, in addition to simply unpacking the software, transfer licenses from update diskettes and run a setup program if one accompanies the software package.

© 2005, QNX Software Systems

Examples:

Install a software package from floppy disks using the drive **/dev/fd0**:

```
/etc/install
```

Unpack and install an update contained in the file **961002.tar.F**:

```
/etc/install -u 961002.tar.F
```

Unpack and install an updated contained in the file **961002.tar.gz**:

```
/etc/install -u 961002.tar.gz
```

Exit status:

0 The installation completed successfully.

>0 The installation was not completed successfully. Either an invalid command-line parameter was supplied, an error occurred during the installation, or the installation was aborted midway.

Caveats:

This utility applies only to software update archives and to packages supplied on floppy diskettes. Software packages supplied on CD-ROM will use a different installation procedure. See the instructions supplied with the CD-ROM for details.

See also:

freeze, **gzip/unzip/zcat**, **license**, **pax**

Provide some of the BIOS INT 10 functions (QNX)

Syntax:

int10 [*options***...] [***command* **| &]**

Options:

-a Auto-detect BIOS types; this is the default. The options **-b**, **-t**, and **-s** override auto-detection; use **-a** to add auto-detected types.

-b *type* Set BIOS type, overriding auto-detection.

-B Start video BIOS from a virtual warm-boot.

-c *console* **| none**
Attach to this console (default is standard error) or don't assume any console.

-E Test EEPROM if applicable.

-KI Allow access to all software interrupts.

-KP Allow access to all I/O ports.

-KV Don't recheck video modes.

-R Display CPU registers before and after INT 10 calls.

-s *script_file* Load this SVPMI script file, overriding auto-detection. A selection of SVPMI scripts is shipped with the operating system under **/usr/lib/svpmi/**.

-t *trans_file* Use this translation file, overriding auto-detection. For a description of translation files see **/etc/readme/technotes/int10**.

-v Be verbose.

command **|** **&** If **int10** isn't run in the background (with **&**), you can specify one of the following optional display commands:

This command:	Displays:
id	current BIOS type
modes	list of supported video modes
query	list of supported BIOS types

Once the information is displayed, **int10** will terminate.

Description:

The **int10** utility provides support for programs using the INT 10 software interrupt. It provides only those INT 10 BIOS functions (as well as the VESA Super VGA BIOS Extension) required by programs that use the Watcom Graphics Library. Once the emulator is running, programs can use the INT 10 software interrupt to change video modes and to set color-palette entries.

If you use **-s**, requests will be satisfied by executing code generated from the SVPMI script. Otherwise, **int10** will execute the BIOS.

When programs using INT 10 calls terminate, the **int10** utility attempts to restore the previous video mode.

To use Watcom Graphics Library functions, you must start **int10** in the background.

Examples:

Auto-detect the video BIOS and display the list of modes:

```
int10 modes
```

Auto-detect the video BIOS and install the INT 10 service:

```
int10 &
```

Load an SVPMI script and install the INT 10 service:

```
int10 -s vga &
```

Caveats:

Since **int10** interprets the BIOS code, this code runs slower than it would if it were running natively. Because of this speed delay, some BIOSs with tight timing restrictions may fail to work with **int10**. If this happens, you'll need to use the **-s** option.

See also:

int10 technical note (available on QUICS)

Watcom C — Graphics Library Reference

Iso9660fsys

© 2005, QNX Software Systems

ISO 9660 Filesystem Manager (QNX)

Syntax:

Iso9660fsys [-w *secs***] [***directory***=***file***]... &**

Iso9660fsys [-s] [-w *secs***] &**

Options:

-s	Suppress scanning of the **/dev** directory for CD-ROM block special files.
-w *secs*	When mounting the filesystem, wait no more than this many seconds. This option is useful if you want to make **Iso9660fsys** wait for the **Fsys** driver to set up its devices. If you don't specify this option, **Iso9660fsys** will wait no more than 60 seconds for the devices to be set up.
*directory***=***file*	The directory where the ISO 9660 filesystem is to be mounted and a file containing the filesystem. If you use this syntax, no scan of the **/dev** directory is performed. The file you specify doesn't have to be a CD-ROM block special file. For example, it could be a regular file that's an image of a CD-ROM or it could be a WORM drive containing an ISO 9660 filesystem.

Description:

The **Iso9660fsys** filesystem manager provides transparent access to ISO 9660 filesystems. It supports Rock Ridge Interchange Protocol (RRIP) extensions (for a POSIX filesystem) and also supports High Sierra filesystems.

Iso9660fsys provides this functionality on top of **Fsys** or **Blkfsys** with their drivers which provide access to the actual CD-ROM drive through block special files in the **/dev** directory.

By default, **Iso9660fsys** does a scan until at least one CD-ROM block special file is found or the timeout occurs. All CD-ROM devices found during the scan are mounted. If you specify **-s** or the *directory*=*file* syntax, the scan isn't performed.

CAUTION:

If you use Rundos (deprecated) to access the mounted filesystem, Rundos will ignore the RRIP filename extensions in favor of the ISO filenames. It will also ignore filenames that don't comply with ISO 9660 level 1.

For *directory*=*file*, you should also note that the specified file needn't be an actual CD-ROM as long as it contains an ISO 9660 filesystem.

Information directory

A directory called **.cd.information.** is always created beneath the directory of the mounted ISO filesystem (e.g. **/cd0/.cd.information.**). Another directory may also be created if a valid volume ID is present on that filesystem. The name of this other directory is **=***volume_id* and may contain up to 11 characters.

Both of these directories contain the same files. These files contain information about the mounted filesystem. If any of the information fields described in the following table are empty, the corresponding file of the same name won't be present.

Field/File	Contents
creation	zero-length file with a time stamp of the creation date
modification	same as above, but for modification date
expiration	same as above, but for expiration date
effective	same as above, but for effective date

continued...

Field/File	Contents
volumesetid	Volume set ID
publisher	CD-ROM publisher
preparer	CD-ROM preparer
application	CD-ROM application
copyright	CD-ROM copyright information
abstract	CD-ROM abstract
bibliographic	CD-ROM bibliographic (not on High Sierra CDs)
system	CD-ROM system area description
volumeid	CD-ROM volume ID (used to make the "=" directory)
rrip_version	version of Rock Ridge Interchange Protocol
control	QNX control file (described below)

Control file

The control file is always present. By writing commands to this special file, you can control the CD-ROM drive. For example, when it is supported by the hardware, the following command would eject the disk from the drive:

```
echo eject >/cd0/.cd.information./control
```

Note that commands must be written atomically to this file. The following table lists the commands you can use.

Command	Description
raw=on|off	display raw ISO 9660 filenames (default is off); if **raw** is turned **on**, ISO 9660 filenames won't be translated by **Iso9660fsys**
rockridge=on|off	enable Rock Ridge extensions (default is **on**)
uid=*user_id*	use this user ID for ISO 9660 filenames (default is 0)
gid=*group_id*	use this group ID for ISO 9660 filenames (default is 0)
play[**=***first*[**-***last*]]	play audio CD from *first* to *last* track
volume=*level*	adjust the volume (0-255)
stop	stop audio CD from playing
pause	pause audio CD
resume	resume audio CD
eject	eject CD from drive
load	reload CD to drive

The **play**, **volume**, **stop**, **pause**, **resume**, **eject**, and **load** commands are ignored if they're not supported by the drive. If you're using a SCSI drive and it doesn't always respond, you might wish to verify that your drive supports the following SCSI commands, which are the actual directives used by the SCSI driver.

Function	SCSI type	Command name	Command code
play	SCSI2	**PLAY AUDIO TRACK/INDEX**	0x48

continued...

Function	SCSI type	Command name	Command code
volume	SCSI2	**MODE SELECT**	0x15
stop	SCSI	**START STOP UNIT**	0x1B
eject	SCSI2	**START STOP UNIT**	0x1B
load	SCSI2	**START STOP UNIT**	0x1B
pause	SCSI2	**PAUSE/RESUME**	0x4B
resume	SCSI2	**PAUSE/RESUME**	0x4B

Examples:

Scan the **/dev** directory and automatically mount any CD-ROM block special files. If the scan were to find **/dev/cd0**, the name of the newly mounted ISO filesystem would be **/cd0**:

```
Iso9660fsys &
```

Mount the block special file **/dev/cd0** as **/cdrom**; don't scan for other CD-ROM devices:

```
Iso9660fsys /cdrom=/dev/cd0 &
```

Files:

Iso9660fsys closes its standard input, standard output and standard error immediately after successfully performing its startup initializations. If errors are encountered on startup, a diagnostic message is written to standard error.

Iso9660fsys adopts portions of the pathname prefix tree under which the files in the mounted ISO9660 filesystem will be found. By default, this mount point will be named */name* where *name* is the basename of the block special file under **/dev** the filesystem is contained in. This mount point may be overridden by the specification of *directory***=***file* command-line arguments.

Exit status:

Iso9660fsys will normally never terminate. However, it may exit if an error is encountered during its startup initializations or if it has not been run in the background and therefore forks another copy of itself allowing the original to exit.

0 **Iso9660fsys** was not started with an ampersand (&) on the command line and has successfully forked another copy to the background as a process group leader.

>0 An error occurred during startup. A diagnostic message will have been written to the standard error.

Caveats:

Some CD-ROM vendors may recommend the following alternative method of mounting an ISO 9660 filesystem:

mount -t iso9660 *block_special_file directory*

For this command to succeed, **Iso9660fsys** must be running in the background. For example, the following commands would mount the ISO filesystem as **/cd0**:

```
Iso9660fsys -s &
mount -t iso9660 /dev/cd0 /cd0
```

See also:

Blkfsys, **Dosfsys**, **Fsys**, **Fsys.***, **mount**, **umount**

join

© 2005, QNX Software Systems

Relational database operator (POSIX)

Syntax:

join [-a *file_number***] [-e** *string***] [-j**n *field***]**
[-o *list***] [-t** *char***] [-v** *file_number***] [-1** *n***] [-2** *n***]**
file1 file2

Options:

Options **-a**, **-j**, and **-v** use the *file_number* argument. Specify **1** to refer to *file1* or **2** to refer to *file2*.

-1 *n* (the number one) Join on the *n*th field of *file1*. Fields are numbered, starting with 1.

-2 *n* Join on the *n*th field of *file2*. Fields are numbered, starting with 1.

-a *file_number*

In addition to the default output, produce a line for every unpairable line in file *file_number*. If both **-a 1** and **-a 2** are specified, both sets of information are output, and information for **-a 2** is always printed first.

-e *string* Replace empty output fields by the string *string*.

-jn *field* (Obsolescent) Join on the specified field of file *n*. If *n* is neither **1** nor **2** (e.g. **-j0 3**), use the specified field in both files. Fields are numbered starting with **1**.

It is preferable to use options **-1** and **-2** in place of option **-j***n*.

-o *list* Produce output in which each line comprises the fields specified in *list*. Each entry in *list* has the form:

*file_number***.***field*

where *field* is a field number. Each entry can be separated by a space or a comma.

Output is written only for lines with matching join fields. The join field isn't written unless it's included in *list*.

-t *char* Use the character *char* as a separator, for both input and output. Every appearance of *char* in a line is significant. When this option is specified, the collating sequence should be the same as that produced by **sort**, *without* the **-b** option.

-v *file_number*

Instead of the default output, produce a line only for every unpairable line in *file_number*.

file1 file2 The names of the text files. If either *file1* or *file2* is **-**, the standard input is used.

Description:

The **join** utility forms a "join" of the two relations specified by the lines of *file1* and *file2*. The join is written to the standard output.

The files *file1* and *file2* are compared on the basis of a "join field" found in both files. For every pair of lines in *file1* and *file2* that have identical join fields, **join** prints one output line. The output line normally consists of the join field, followed by the rest of the line from *file1* and then the rest of the line from *file2*. By default, the join field is the first field in each line.

Both *file1* and *file2* should be sorted in an increasing collating sequence on their join fields (i.e. the same sequence performed by **sort -b**). Otherwise, some field matches may not be reported. Note, however, that when option **-t** is specified, the collating sequence should be the same as that produced by **sort**, *without* the **-b** option.

The default input field separators are **blank**s. In this case, multiple separators count as one field separator; leading separators are ignored. The default output field separator is a **space**.

© 2005, QNX Software Systems

Examples:

Join the password file and group file, matching on the numeric group ID and outputting the login name, group name, and login directory. It's assumed that the files have been sorted in collating sequence on the groupid fields.

```
join -1 4 -2 3 -o 1.1 2.1 1.6 -t:  /etc/passwd /etc/group
```

Exit status:

0 All input files were output successfully.

>0 An error occurred.

See also:

awk, **comm**, **sort**, **uniq**

kbd

Specify keyboard layout (QNX)

Syntax:

kbd *kbfile*

Options:

kbfile The name of a keyboard layout file (typically the name of a country). You must specify the full pathname.

Description:

The **kbd** utility is used to configure the QNX keyboard driver to respond to a particular keyboard layout. The default keyboard layout is the United States configuration. Users of QNX in other countries will use the **kbd** utility at system initialization time to specify some other keyboard layout.

Examples:

Use the German keyboard layout:

```
kbd /etc/config/kbd/Germany
```

Use a custom keyboard layout:

```
kbd /etc/config/kbd/custom
```

Files:

/etc/config/kbd/

Keyboard layout files are typically found in this directory. Custom keyboard layouts can also be used by specifying the name of a custom keyboard file.

© 2005, QNX Software Systems

See also:

kedit

kedit

Edit a keyboard layout file (QNX)

Syntax:

kedit [-o *file***]** *kbfile*

Options:

-o *file* Write output to the specified file if the user requests to save changes when exiting.

kbfile A keyboard layout file, usually found in the **/etc/config/kbd** directory. You must specify the full pathname. The special name **system** can be used to edit the currently defined layout.

Description:

The **kedit** utility is a fullscreen program that lets you generate custom keyboard layouts. The program displays several screens that let you view or change the codes assigned to individual keys. There's a separate screen for each of these modes:**UNSHIFTED**, **SHIFTED**, **CTRL**, **ALT**.

Once you've entered **kedit**, you can perform the following actions:

In order to:	Press:
select a keyboard layout	Tab
toggle through shift modes (see list above)	PgUp *or* PgDn
select a key	← *or* →
change the character assigned to a key	↑ *or* ↓
make a key CapsLock-dependent	F1
make a key NumLock-dependent	F2
make a key a "dead" key	F3
exit **kedit**	Esc

You can determine that a key will generate a different character if either CapsLock or NumLock is active. The value of the opposite shift state will be sent instead of the default value. For example, if CapsLock is active on a US keyboard, pressing A will generate A instead of a, and if both CapsLock and Shift are active, A will generate a, not A. Typically, the A-Z keys are CapsLock-dependent, while the keys on the numeric keypad are NumLock-dependent.

Dead keys, which are useful for non-US layouts (e.g. French, German), don't generate a character when used alone. Instead, they're used to modify the effects of other keys. The user presses the dead key, then presses the key that it modifies. Dead keys are often used for creating accented characters. For example, to create the umlaut character, the user presses the umlaut dead key, followed by Shift – U.

If you change any key assignments, and then attempt to exit by pressing Esc, you'll be asked whether the changes should be saved.

Examples:

Edit the German layout file:

```
kedit /etc/config/kbd/Germany
```

Create a new layout file based on the current keyboard layout:

```
kedit -o /etc/config/kbd/custom system
```

Files:

/etc/config/kbd

Keyboard layout files are typically found in this directory.

See also:

kbd

Syntax:

kill [-n *nid***] [-***signal_name* **|** **-***signal_number***]** *pid***...**

kill -l

Options:

-*signal_name*	The name of the signal to be sent to the specified process. Values of *signal_name* are recognized in case-independent fashion, with or without the **SIG** prefix.
-*signal_number*	(Deprecated) A non-negative decimal integer representing the signal to be sent to the specified process.
-l	("el") Don't send signals. List possible values for *signal_name*.
-n *nid*	Kill the process on the specified node. This option is not available in the shell builtin version of **kill**.
pid	A decimal integer specifying a process or process group to be signaled. A positive number *pid* is simply a process ID. A negative number *pid* has its absolute value taken as a process group ID. The signal is sent to all processes belonging to the group. A *pid* of zero sends the signal to all processes owned by the user in the current shell's process group.

© 2005, QNX Software Systems

WARNING:

Don't use a *pid* of zero when logged on as `root`. Signalling all the background processes owned by superuser and any current superuser foreground processes (e.g. backups in progress) may produce unpredictable results.

Description:

The **`kill`** utility sends a signal to the process(es) specified by each *pid* operand. By default, **`kill`** sends the **`SIGTERM`** signal, but you may override this default by naming a signal to be sent.

To print out the list of signals that may be sent, use **`kill`** with the **`-l`** option:

```
kill -l
```

The **`kill`** command is available both as a standalone utility and as a shell builtin. To make sure that you're using the utility, specify the full path.

Examples:

Any of the commands:

```
kill -9 100 -16

kill -sigkill 100 -16

kill -KILL 100 -16
```

sends the **`SIGKILL`** signal to the process whose process ID is 100 and to all processes whose process group ID is 16, assuming the sending process has permission to send that signal to the specified processes, and that they exist.

Exit status:

0 At least one matching process was found for each *pid* option, and the specified signal was successfully sent to each matching process.

>0 An error occurred.

Caveats:

Some shells include a builtin **kill** command. To make sure you're using the **kill** utility, specify the the full path to it.

See also:

sh, **slay**

kill_vcs

© 2005, QNX Software Systems

Kill all virtual circuits to a node (QNX)

Syntax:

kill_vcs *node_number*

Options:

node_number node number of the node you wish to kill all VCS to

Description:

WARNING: This utility has potentially dangerous side-effects — read the complete description!

The **kill_vcs** utility allows a **root** user to eliminate all virtual circuits between the machine the command is run on and the specified node.

Running this utility can also cause the destruction of virtual circuits to nodes other than the one specified!

The **kill_vcs** utility kills virtual circuits by first masking the target node in the local node's netmap table. The utility then tightens the local netpoll values to cause the virtual circuit to be torn down by virtual circuit polling (the polls can't reach the destination because there is no network mapping for it, so they fail). The utility waits approximately four seconds to give **Proc** time to realize that the remote node is not responding, and to tear down the virtual circuit(s). The netpoll parameters are then restored to their previous setting, and the netmap mapping is unmasked.

Note that after waiting for four seconds the network mapping will be unmasked even if it had been masked before **kill_vcs** was run.

Since the network polling parameters affect all virtual circuits to all nodes (not just the node specified), the tightening of the polling parameters may cause VCs to other nodes to be torn down. This will occur if there is a problem polling any of the VCs to other nodes

during the four second time period in which the poll parameters are tightened. In particular, caution should be exercised if there are any virtual circuits open to other nodes through **Net.fd** or any other slow or unreliable interface.

The network mapping and the netpoll parameters are not locked during the duration of **kill_vcs**. It is possible for **kill_vcs** to fail to eliminate virtual circuits if another program (such as **netmask** or **netpoll**) restores the network mapping or changes the network polling parameters before the four second time period has elapsed.

Examples:

Kill all the VCs to node 95:

```
kill_vcs 95
```

Exit status:

>0	An error occurred in locating the local **Net** manager, changing the network mapping, or changing the network polling parameters.
0	The manipulations of the network mapping and network polling parameters succeeded. This success and 0 exit status is not a guarantee that the intended virtual circuits have been torn down.

See also:

netmap, **netpoll**

ksh

© 2005, QNX Software Systems

Korn Shell command interpreter (UNIX)

Syntax:

ksh [-aefhkmnquvx] [*command_file* **[***argument***...]]**

ksh -c [-aefhkmnquvx] *command_string*
[*command_name* **[***argument***...]]**

ksh -i [-aefhkmquvx] [*command_file* **[***argument***...]]**

ksh -s [-aefhkmnquvx] [*argument***...]**

Options:

See **sh**.

Description:

The standard QNX shell, **/bin/sh** is a modified Korn shell derived from a version **pdksh** which was modeled after **ksh86**. A link to **sh** called **ksh** is installed in **/bin** when QNX is installed.

See the documentation for **sh** for documentation on the features of the shell.

See also:

esh, **sh**, **uesh**

Syntax:

less [-[+]aBcCdeEfimMnNqQrsSuUw] [-b *n***] [-x** *n***]**
[-[z] *n***] [-h** *n***] [-j** *n***] [-p** *pattern***]**
[-y *n***] [-[oO]** *logfile***] [-t** *tag***]**
[-T *tagsfile***] [+***cmd***] [++***cmd***] [***file***...]**

Options:

Most options may be changed while **less** is running, via the dash (**-**) command.

Options are also taken from the **LESS** environment variable. The environment variable is parsed before the command line, so command-line options override the **LESS** environment variable. If an option appears in the **LESS** environment variable, it can be reset to its default on the command line by using the two-character combination **-+** at the beginning of the command line.

A dollar sign (**$**) may be used to signal the end of an option string. This is important only for options like **-t** which take a following string.

-? Display a summary of the commands accepted by **less** (the same as the **h** command). If this option is given, all other options are ignored, and **less** exits after the help screen is viewed. (Depending on how your shell interprets the question mark, it may be necessary to quote the question mark, as follows: **-\?**)

-a Start searches after the last line displayed on the screen, thus skipping all lines displayed on the screen. By default, searches begin at the second line on the screen (or after the last found line; see the **-j** option).

-b *n* Use a nonstandard number of buffers. Buffers are 1K, and by default 10 buffers are used (except if data is coming from standard input; see the **-B** option). The number *n* specifies the number of buffers to use.

© 2005, QNX Software Systems

-B Disable automatic allocation of buffers, so that only the default number of buffers is used. If more data is read than will fit in the buffers, the oldest data is discarded. By default, when data is coming from standard input, buffers are allocated automatically as needed to avoid loss of data.

-c Do fullscreen redraws from the top line down. By default, fullscreen redraws are done by scrolling from the bottom of the screen.

-C Clear the screen, then do fullscreen redraws from the top line down. By default, fullscreen redraws are done by scrolling from the bottom of the screen.

-d Suppress the error message normally displayed if the terminal is dumb (i.e. lacks some important capability, such as the ability to clear the screen or scroll backward). The **-d** option does not otherwise change the behavior of **less** on a dumb terminal.

-e Automatically exit the second time end-of-file is reached. By default, the only way to exit **less** is via the **q** command.

-E Automatically exit the first time end-of-file is reached. By default, the only way to exit **less** is via the **q** command.

-f Force nonregular files to be opened (a nonregular file is a directory or a device special file). Also, suppress the warning message when a binary file is opened. By default, **less** will refuse to open nonregular files.

-h *n* Don't scroll backward any more than *n* lines. If it's necessary to scroll backward more than *n* lines, the screen is redrawn in a forward direction instead. (If the terminal can't scroll backward, **-h 0** is implied.)

-i Ignore case; uppercase and lowercase are considered identical. Also, text that is overstruck or underlined can be searched for. This option is ignored if any uppercase letters appear in the search pattern.

-j *n* Use this line on the screen to position "target" lines. Target lines are the object of text searches, tag searches, jumps to a line number, jumps to a file percentage, and jumps to a marked position. The screen line is specified by a number: the top line on the screen is **1**, the next is **2**, and so on. The number may be negative to specify a line relative to the bottom of the screen: the bottom line on the screen is **-1** (the number one), the second to the bottom is **-2**, and so on. With the **-j** option, searches begin at the line immediately after the target line. For example, if **-j4** is used, the target line is the fourth line on the screen, so searches begin at the fifth line on the screen.

-m Prompt verbosely (like **more**), with the percent into the file. By default, **less** prompts with a colon.

-M Prompt even more verbosely than **more**.

-n Suppress line numbers. The default (to use line numbers) may cause **less** to run more slowly in some cases, especially with a very large input file. Suppressing line numbers with **-n** will avoid this problem. Using line numbers means: the line number will be displayed in the verbose prompt and in the **=** command; the **v** command will pass the current line number to the editor.

-N Display a line number at the beginning of each line.

-o *file* Copy input to the named file as it is being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, **less** will ask for confirmation before overwriting it.

© 2005, QNX Software Systems

-O *file*
: Copy input to the named file as it is being viewed. This applies only when the input file is a pipe, not an ordinary file. If the file already exists, **less** will *NOT* ask for confirmation before overwriting it.

  If no log file has been specified, the **-o** and **-O** options can be used from within **less** to specify a log file. Without a filename, they will simply report the name of the log file.

-p *pattern*
: Start at the first occurrence of *pattern* in the file. The **-p** option on the command line is equivalent to specifying **+/***pattern*.

-q
: Be moderately quiet; don't ring the terminal bell if an attempt is made to scroll past the end of the file or before the beginning of the file. If the terminal has a "visual bell," it is used instead. The bell will be rung on certain other errors, such as typing an invalid character. The default is to ring the terminal bell in all such cases.

-Q
: Be totally quiet; never ring the terminal bell.

-r
: Display "raw" control characters. The default is to display control characters using the caret (^) notation. For example, a **Ctrl-A** (001 octal) is displayed as **^A**.

  Note that when **-r** is used, **less** can't keep track of the actual appearance of the screen (since this depends on how the screen responds to each type of control character). Thus, various display problems may result, such as long lines being split in the wrong place.

-s
: Squeeze consecutive blank lines into a single blank line.

-S
: Chop rather than fold lines longer than the screen width. That is, the remainder of a long line is simply discarded. The default is to fold long lines; that is, display the remainder on the next and subsequent lines.

-t *tag* Edit the file containing the specified tag. For this to work, there must be a file called **tags** in the current directory. (See **ctags**.)

-T *tagsfile* Use the specified tags file in place of the **tags** file.

-u Treat backspaces and carriage returns as printable characters (i.e. they're sent to the terminal when they appear in the input).

-U Treat backspaces and carriage returns as control characters (i.e. they're handled as specified by the **-r** option).

By default, if neither **-u** nor **-U** is given, backspaces that appear adjacent to an underscore character are treated specially: the underlined text is displayed using the terminal's hardware underlining capability. Backspaces that appear between two identical characters are also treated specially: the overstruck text is printed using the terminal's hardware boldface capability. Other backspaces are deleted, along with the preceding character. Carriage returns immediately followed by a newline are deleted; other carriage returns are handled as specified by the **-r** option.

-w Use blank lines to represent lines past the end of the file. By default, a tilde (~) character is used.

-x *n* Set tab stops every *n* positions (the default 8).

-y *n* Specifies a maximum number of lines to scroll forward. If it's necessary to scroll forward more than *n* lines, the screen is repainted instead. The **-c** or **-C** option may be used to repaint from the top of the screen if desired. By default, any forward movement causes scrolling.

-[**z**] *n* Change the default scrolling window size to *n* lines. The default is one screenful. The **z** and **w** commands

© 2005, QNX Software Systems

can also be used to change the window size. Note that the **z** may be omitted (as in **-***n*).

+*cmd* — If a command-line option begins with **+** (plus), the remainder of that option is taken to be an initial command to **less**. For example, **+G** tells **less** to start at the end of the file rather than the beginning; and **+/xyz** tells it to start at the first occurrence of "**xyz**" in the file.

As a special case, **+***number* acts like **+***number***g**; that is, it starts the display at the specified line number (note, however, that this may be slow — see the **g** command).

++*cmd* — If the option starts with two plus signs (**++**), the initial command applies to every file being viewed, not just the first one. The **+** command (described in the "Commands" section) may also be used to set or change an initial command for every file.

file — A pathname of an input file. If no *file* operands are specified, **less** uses the standard input. If a *file* operand is the dash character (**-**), the standard input is read at that point of the sequence.

Description:

The **less** utility is a program similar to **more**, but **less** allows backward movement in the file as well as forward movement. The **less** utility uses the system terminal capability database, so it can run on a variety of terminals. There is limited support for hardcopy terminals (on a hardcopy terminal, lines that should be printed at the top of the screen are prefixed with **up-arrow**).

The **less** utility will display a screen full of information, then prompt for user input by displaying a : prompt at the bottom of the screen. Commands can then be entered from the keyboard.

Commands

Commands may be preceded by a decimal number, called *n* in the following descriptions. The number is used by some commands, as indicated.

Command:	Description:
h	Help: display a summary of these commands. If you forget all the other commands, remember this one.
[*n*] spacebar *or* [*n*] **f**	Scroll forward *n* lines; default is one window (see the **-z** option). If *n* is more than the screen size, only the final screenful is displayed.
[*n*] **z**	Like spacebar, but if *n* is specified, it becomes the new window size.
[*n*] Enter	Scroll forward *n* lines; the default is 1. The entire *n* lines are displayed, even if *n* is more than the screen size.
[*n*] **d**	Scroll forward (down) *n* lines; the default is 1/2 screen size. If *n* is specified, it becomes the new default for subsequent **d** and **u** commands.
[*n*] **b**	Scroll backward *n* lines; the default is one window (see the **-z** option). If *n* is more than the screen size, only the final screenful is displayed.
[*n*] **w**	Like **b**, but if *n* is specified, it becomes the new window size.
[*n*] **k**	Scroll backward *n* lines; the default is 1. The entire *n* lines are displayed, even if *n* is more than the screen size.

continued...

Command:	Description:
[*n*] **u**	Scroll backward (up) *n* lines; the default is 1/2 screen size. If *n* is specified, it becomes the new default for subsequent **d** and **u** commands.
[*n*] **r**	Redraw the screen.
[*n*] **F**	Scroll forward, and keep trying to read when the end of the file is reached. Normally this command would be used when already at the end of the file. It is a way to monitor the tail of a file which is growing while it is being viewed. (The behavior is similar to the **tail -f** command.)
[*n*] **g**	Go to line *n* in the file; the default is 1 (beginning of file). (Note that this may be slow if *n* is large.)
[*n*] **G**	Go to line *n* in the file; the default is the end of the file. Note that this may be slow if *n* is large, or if *n* isn't specified and standard input, rather than a file, is being read.
[*n*] **p**	Go to a position *n* percent into the file. *n* should be between 0 and 100. This works if standard input is being read, but only if **less** has already read to the end of the file. This is always fast, but not always useful.

continued...

Command:	Description:
[*n*] **{**	If a left brace (**{**) appears in the top line displayed on the screen, the **{** command will go to the matching right brace, which is positioned on the bottom line of the screen. If there is more than one left brace on the top line, a number *n* may be used to specify the *n*th brace on the line.
[*n*] **}**	If a right brace (**}**) appears in the bottom line displayed on the screen, the **}** command will go to the matching left brace, which is positioned on the top line of the screen. If there is more than one right brace on the top line, a number *n* may be used to specify the *n*th brace on the line.
[*n*] **(**	Like **{**, but applies to parentheses rather than braces.
[*n*] **)**	Like **}**, but applies to parentheses rather than braces.
[*n*] **[**	Like **{**, but applies to square brackets rather than braces.
[*n*] **]**	Like **}**, but applies to square brackets rather than braces.
Esc Ctrl – F *charchar*	Acts like **{**, but uses the two characters as open and close brackets, respectively. For example, **ESC ^F <>** could be used to go forward to the **>** which matches the **<** in the top displayed line.

continued...

© 2005, QNX Software Systems

Command:	Description:
Esc Ctrl – B *charchar*	Acts like **}**, but uses the two characters as open and close brackets, respectively. For example, **ESC ^B <>** could be used to go backward to the **<** which matches the **>** in the bottom displayed line.
m*char*	Followed by any lowercase letter, marks the current position with that letter.
'*char*	(single quote) Followed by any lowercase letter, returns to the position that was previously marked with that letter. Followed by another single quote, returns to the position at which the last "large" movement command was executed. Followed by a **^** or **$**, jumps to the beginning or end of the file respectively. Marks are preserved when a new file is examined, so the **'** command can be used to switch between input files.
[*n*] **/***pattern*	Search forward in the file for the *n*th line containing the pattern; *n* defaults to 1. The pattern is a regular expression, as recognized by **sed**. The search starts at the second line displayed (but see the **-a** and **-j** options, which change this). Certain characters are special if entered at the beginning of the pattern; see note at end of table.

continued...

Command:	Description:
[*n*] **?***pattern*	Search backward in the file for the *n*th line containing the pattern. The search starts at the line immediately before the top line displayed. Certain characters are special as in the **/** command; see note at end of table.
[*n*]Esc **/***pattern*	Same as **/***.
[*n*]Esc **?***pattern*	Same as **?***.
[*n*] **n**	Repeat previous search, for *n*th line containing the last pattern. If the previous search was modified by **!**, the search is made for the *n*th line *not* containing the pattern. If the previous search was modified by *****, the search continues in the next (or previous) file if not satisfied in the current file. There is no effect if the previous search was modified by **@**.
[*n*] **N**	Repeat previous search, but in the reverse direction.
Esc **n**	Repeat previous search, but crossing file boundaries. The effect is as if the previous search were modified by *****.
Esc **N**	Repeat previous search, but in the reverse direction and crossing file boundaries.

continued...

© 2005, QNX Software Systems

Command:	Description:
:e [*filename*]	Examine a new file. If the filename is missing, the "current" file (see the **:n** and **:p** commands) from the list of files in the command line is re-examined. A percent sign (**%**) in the filename is replaced by the name of the current file. A pound sign (**#**) is replaced by the name of the previously examined file. The filename is inserted into the command-line list of files so that it can be seen by subsequent **:n** and **:p** commands. If the filename consists of several files, they are all inserted into the list of files and the first one is examined.
E	Same as **:e**.
[*n*] **:n**	Examine the next file (from the list of files given in the command line). If a number *n* is specified, the *n*th next file is examined.
[*n*] **:p**	Examine the previous file in the command-line list. If a number *n* is specified, the *n*th previous file is examined.
[*n*] **:x**	Examine the first file in the command-line list. If a number *n* is specified, the *n*th file in the list is examined.
=	Print some information about the file being viewed, including its name and the line number and byte offset of the bottom line being displayed. If possible, this also prints the length of the file, the number of lines in the file, and the percentage of the file above the last displayed line.

continued...

Command:	Description:
-*char*	Followed by one of the command-line option letters, this will change the setting of that option and print a message describing the new setting. If the option letter has a numeric value (such as **-b** or **-h**), or a string value (such as **-P** or **-t**), a new value may be entered after the option letter. If no new value is entered, a message describing the current setting is printed and nothing is changed.
-+*char*	Followed by one of the command-line option letters, this will reset the option to its default setting and print a message describing the new setting. The **-+X** command does the same thing as **-+X** on the command line. This doesn't work for string-valued options.
--*char*	Followed by one of the command-line option letters, this will reset the option to the *opposite* of its default setting and print a message describing the new setting. The **--X** command does the same thing as **-X** on the command line. This doesn't work for numeric or string-valued options.
+*cmd*	Causes the specified *cmd* to be executed each time a new file is examined. For example, **+G** causes **less** to initially display each file starting at the end rather than the beginning.
V	Print the version number of **less** being run.
q	Exit **less**.

continued...

© 2005, QNX Software Systems

Command:	Description:
v	Invoke an editor to edit the current file being viewed. The editor is taken from the environment variable **EDITOR**, or defaults to **vedit**.
! *shell_command*	Invoke a shell to run the *shell_command* given. A percent sign (**%**) in the command is replaced by the name of the current file. A pound sign (**#**) is replaced by the name of the previously examined file. **!!** repeats the last shell command. **!** with no shell command simply invokes a shell. In all cases, the shell is taken from the environment variable **SHELL**, or defaults to **sh**.
| *m shell_command*	*m* represents any mark letter. Pipes a section of the input file to the given shell command. The section of the file to be piped is between the current position and the position marked by the letter. *m* may also be **^** or **$** to indicate beginning or end of file, respectively. If *m* is a dot (**.**) or newline, the current screen is piped. The current screen is the minimum amount piped in any case.

Certain characters are special if entered at the beginning of the pattern for the **/** and **?** commands; they modify the type of search rather than become part of the pattern:

! Search for lines that do *not* match the pattern.

***** Search multiple files. That is, if the search reaches the end of the current file without finding a match, the search continues in the next file in the command-line list.

@ Begin the search at the first line of the first file in the command-line list, regardless of what is currently displayed on the screen or the settings of the **-a** or **-j** options.

Files:

/usr/local/lib/less.hlp

Help file for **less**'s **help** command.

Environment variables:

COLUMNS	Sets the number of columns on the screen. Takes precedence over the number of columns specified by the **TERM** variable.
EDITOR	The name of the editor (used for the **v** command).
LESS	Flags that are passed to less automatically.
LESSEDIT	Editor prototype string (used for the **v** command).
LINES	Sets the number of lines on the screen. Takes precedence over the number of lines specified by the **TERM** variable.
SHELL	The shell used to execute the **!** command, as well as to expand filenames.

© 2005, QNX Software Systems

TERM The type of terminal on which **less** is being run.

Contributing author:

Mark Nudelman

Caveats:

The **=** command reports the line number of the line at the top of the screen, but the byte and percent of the line at the bottom of the screen.

If the **:e** command is used to name more than one file and if one of the named files has been viewed previously, the new files may be entered into the list in an unexpected order.

See also:

ctags, **more**, **vi**

Syntax:

lex [-Icdflnoitv78] [-s *skeleton***] [***file***...]**

Options:

-7

-8 Generate 7- or 8-bit scanners. The 7-bit scanner is the standard for most **lex** systems. This implementation defaults to 8-bit scanners, but if you intend to use the generated scanner on other systems, you should use **-7** to ensure maximum portability.

-c Generate a C language scanner (default). This option has no real effect since **lex** generates only C programs, but is required by POSIX to emphasize that the scanner is generated in C.

-d Generate code for debugging the scanner. The code displays what state the scanner is in, what characters are currently expected, etc. Useful for debugging scanners.

-f Generate larger, faster, full-table scanners. Since tables aren't compressed, you shouldn't use this option for large scanners.

-i Generate a case-insensitive scanner.

-I Generate an interactive scanner. An interactive scanner must be able to stop looking ahead as soon as a rule has been matched. If your program is interactive, you should use this option, even though it disables some of the optimizations **lex** performs.

-l ("el") Disable generation of "line directives" in the scanner. This option may be required to step through the scanner using a source-level debugger.

-n Don't generate scanner statistics (default).

© 2005, QNX Software Systems

-o	Minimize the size of the scanner tables, possibly at the cost of execution time.
-s *skeleton*	Use the specified skeleton file (default is **/usr/lib/lex/skeleton**).
-t	Write the scanner to standard output rather than to the **lex.yy.c** file.
-v	Print a summary of the scanner statistics.
filename	The source filename, or standard input if not specified.

Description:

The **lex** utility, which is a modified version of Vern Paxton's *flex 2.3.7*, generates a C program from **lex** source code. The program generated is suitable for processing character input and is designed to interface to the **yacc** utility.

The **lex** utility requires the file **/usr/lib/lex/skeleton**. You may create a link to **lex** called **flex**, which supports a greater set of options for controlling the generated scanner. If you create this link, you must also copy or link the file **/usr/lib/lex/skeleton** to **/usr/local/lib/flex.skel**.

The **lex** specification contains a set of regular expressions and actions where the scanner will perform the action when the regular expression is recognized in the input stream.

The following "publics" are defined in the scanner:

Variable	Description
char ***_yytext_;**	When a token is recognized in the input stream, it is placed into this string, with a terminating \0. In some traditional implementations, this is **char** *yytext*[].
int *yyleng***;**	This integer contains the length of the string *yytext*.
FILE* *yyout***;**	The file written to for default, or **ECHO** actions.
FILE ***_yyin_;**	The file associated with the current input.

The following "externals" are required by the scanner:

Variable	Description
int *input***()**	Return the next character from the input, or return zero to signify no more input.
int *output***(int** *c***)**	Print the char *c* on *yyout*.
int *unput***(int** *c***)**	Return the character *c* to the input stream so it will be read by a subsequent call to the *input*() function.
int *yywrap***()**	Called when **lex** reaches the end of file. If there are further files to be processed, *yywrap()* should initialize *yyin* and return 0, otherwise return 1.

These routines are engineered for simplifying the most common uses of **lex** scanners and may be overridden by defining them in programs themselves.

Variations of the externals are defined in the **lex** library.

© 2005, QNX Software Systems

You can use the following lexical analyzer in conjunction with the parser in **yacc** to implement a simple calculator. You'll find the source (**calc.l**) for this analyzer in **/usr/demo/src**.

```
%{
/*
 * Sample lexical analyzer for a simple calculator.
 *
 */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <math.h>
#define YYSTYPE double
#include "y.tab.h"

extern int lineno;        /* current line number  */
extern YYSTYPE yylval;    /* value of numeric token */

%}

      digit [0-9]
space [ \t]

%%
{space}         { ; }    /* spaces are ignored */

{digit}+\.?|{digit}*\.{digit}+ {
                   yylval = strtod(yytext,0);
                   return NUMBER; }

\*\*            { return '^';  }
last            { return LAST; }
cos             { return COS;  }
exp             { return EXP;  }
sin             { return SIN;  }
sqrt            { return SQRT; }
tan             { return TAN;  }
pi              { yylval = atan(1.0)*4;
                  return NUMBER; }
e               { yylval = exp(1.0);
                  return NUMBER; }
\n              { lineno++; return '\n'; }

.               { return yytext[0]; }

%%
```

Examples:

The following **lex** program will trim trailing **space**s from lines, fold multiple **space**s into a single **tab**, and convert all lowercase text to uppercase:

```
% {
#include <stdio.h>
#include <ctype.h>
% }
% %
[a-z] putchar(toupper(yytext[0]));
[  ]+$
[  ]+  putchar('\t');
```

Files:

lex.yy.c By default **lex** will create this file and put the generated scanner source code in it.

/usr/lib/lex/skeleton

Default skeleton file (**-s** to change).

Author:

Vern Paxton, The University of California at Berkeley

Caveats:

Some code will expect *yytext* to be defined as

```
char yytext[];
```

instead of

```
char *yytext;
```

See also:

yacc

A.V. Aho, R. Sethi and J.D. Ullman, *Compilers: Principles, Techniques and Tools*, Addison-Wesley, 1977.

M.E. Lesk and E.Schmidt, “Lex — A Lexical Analyzer Generator,” *Bell Laboratories Computing Science Technical Report # 39*, October 1975.

S.C. Johnson, “Yacc — Yet Another Compiler-Compiler,” *Bell Laboratories Computing Science Technical Report # 32*, July 1978.

J. P. Bennett, *Introduction to Compiling Techniques — A First Course using ANSI C, LEX, and YACC*, McGraw-Hill, 1990.

Syntax:

Install licenses from floppy:

license [-q] /dev/fd*n*

Update in-memory license information from disk or from another node:

license {-r|-R *node***}**

Install or copy old-style licenses:

license [*source* **[***destination***]]**

Options:

-q	Be quiet; don't print any messages.
-r	Read the licenses from the license file **/.licenses** (a file which contains new-style licenses from QNX 4.23 and later), and from the directory **/etc/licenses** (old-style licenses) and place them in this node's in-memory license database.
-R *node*	Read up-to-date licenses from the specified node's in-memory license database.
source	The source of the license file, which can be one of:

© 2005, QNX Software Systems

Source	Description
/dev/fd*n*	A floppy drive that the licenses are transferred from (*n* can be either 0 or 1).
directory	A directory that contains an old-style license files.
file	A specific old-style license file to transfer.

The default is **/dev/fd0**.

destination The pathname of the old-style license directory that is to be updated (default is the **/etc/licenses** directory in the current filesystem).

Description:

The **license** utility transfers license files to your hard disk. These license files are used by the operating system and some applications to verify the licensing of the software. Licenses from QNX version 4.23 and later are kept in the file **/.licenses**. Pre-QNX 4.23 licenses are stored in files under the **/etc/licenses** directory.

When you receive a new license for your network or for an application that is licensed to you, you should place the diskette containing the license files in a floppy drive and run the **license** utility.

You must be logged in as **root** to use **license**.

Before you run the **license** utility to transfer licenses from a diskette, make sure the floppy driver has been started.

The utility will copy the license file from the diskette to your hard disk. Note that the new licenses won't become effective until you reboot or until you execute **license** with either **-R** *node* or **-r**.

Nodes that boot over the network will inherit licenses from the node they boot from.

All machines that boot from hard disk in a local area network should have all network licenses installed. This can be accomplished by using **license** *source destination* for old-style licenses, and **cp** *sourcenode***/.licenses** *destnode***/.licenses** for new-style licenses.

Examples:

Copy the license file from the floppy diskette in drive 0 to the **/.licenses** file or the **/etc/licenses** directory, depending on whether new-style or old-style licenses are on the disk:

```
license
```

or

```
license /dev/fd0
```

Copy the license files from the floppy diskette in drive 0 on node 10 to the **/.licenses** directory or the **/etc/licenses** directory, depending on whether new-style or old-style licenses are on the disk:

```
license //10/dev/fd0
```

Copy all old-style license files from the **/etc/licenses** directory on node 1 to the **/etc/licenses** directory on node 5:

```
license //1/etc/licenses //5/etc/licenses
```

Reload all licenses into memory from disk:

```
license -r
```

Copy new-style licenses from node 1 to node 5:

```
cp //1/.licenses //5/.licenses
```

© 2005, QNX Software Systems

See also:

licinfo

Installation & Configuration

licinfo

List licenses in use (QNX)

Syntax:

licinfo [*options***...]**

Options:

-a	List all licenses even if none are in use.
-n *node*	Check licenses in use on this node only.
-N	Display output by node rather than by license.
-l *license*	("el") Check nodes with this license only.
-v	Be verbose about any failures to obtain information.

Description:

The **licinfo** utility will write information pertaining to the on-line software licenses to the standard output.

You must be logged in as **root** to use **licinfo**.

The output format of **licinfo** depends on the options supplied. By default, it displays information by product (in no particular order) in the following format:

product num_used/num_licenses [*node*...]

but when **-N** is given, the output is displayed by node, each entry in the format:

Node *node* [*product*...]

By default, output is listed for licenses used by any/all nodes on the network. If the **-n** *node* option is specified, only licenses in use by that node are listed. Normally, licenses will be listed only if at least

one is in use. If the **-a** option is given, all available licenses will be listed even if none are currently in use.

Note that node licenses for the operating system itself are not counted as "in use" and do not appear unless the **-a** option is specified. You can also see the number of OS licenses by examining the "Nodes" value returned by **sin info**.

Examples:

Show all licenses in use:

```
licinfo
```

Show all C compiler licenses in use:

```
licinfo -l wcc
```

Files:

The results of the license query are written to the standard output. If errors occur and **-v** is specified, a diagnostic message is written to the standard error. The standard input is not used.

Exit status:

>0 An error occurred.

0 The license information was successfully reported.

See also:

license

Installation & Configuration

Syntax:

ln [-f|-i] [-s] [-v] *source_file target_file*

ln [-f|-i] [-s] [-v] *source_file*... *target_dir*

Options:

-f	Force existing destination pathnames to be removed before linking; don't prompt for confirmation.
-i	(QNX extension) Run interactively; write a prompt to the standard error output requesting confirmation for each link that would overwrite an existing file.
-s	Create a symbolic link.
-v	Verbose. Write actions performed to standard output.
source_file	The pathname of a file to be linked. If **-s** is specified, this file need not exist.
target_file	The pathname of the new directory entry to be created.
target_dir	The pathname of an existing directory in which the new directory entries are to be created.

Description:

The **ln** utility has two syntax forms, as follows:

First syntax form:

ln [**-f|-i**] [**-s**] *source_file target_file*

The **ln** utility creates a new directory entry (link) at the destination path specified by the *target_file* operand which points to the *source_file* as a hard or symbolic link, depending on the **-s** option. This syntax form is assumed when the destination path *does not* name an existing directory.

Second syntax form:

ln [**-f|-i**] [**-s**] *source_file... target_dir*

For each *source_file*, **ln** creates a new directory entry at a destination path in the existing directory named by the *target_dir* operand.

The destination path for each *source_file* is the same as its basename (final path component). For example:

```
ln dir/dir/myfile /existingdir
```

will create **/existingdir/myfile** as a link to **dir/dir/myfile**.

This second syntax form is assumed when either the destination names an existing directory, or when more than one source file is specified.

If the destination path exists and you have write permission for the existing destination file, or if **-f** was specified, **ln** will unlink the destination, then create the new link.

If you don't have write permission for an existing directory path, and **-f** is not specified, and if the standard input is a tty, **ln** will prompt you for confirmation prior to unlinking the existing file. If standard input isn't a tty, **ln** will write a diagnostic message to standard error, and go on to the next *source_file* without unlinking the destination file.

To create the new link, or replace a file with a link, you will need write permissions for the directory in which the new link is going to reside. Note that **root** will always have this permission regardless of the file permission settings.

Hard links are limited to within the same filesystem as the original file and aren't permitted on directories. With symbolic links, however, you can link *any* pathname to a file. A symbolic link is a special file that has the destination pathname as its data. For more information, see the section on symbolic links in QNX *System Architecture*.

Examples:

Create a link to **/home/curious/monkey** called **gorilla** in the **/home/george** directory:

```
ln /home/curious/monkey /home/george/gorilla
```

Create a symbolic link to the directory **//1/home/fred** called **//2/home/barney**:

```
ln -s  //1/home/fred //2/home/barney
```

Exit status:

0	All the specified files were linked successfully.
>0	An error occurred.

Caveats:

CAUTION: If a destination path exists, and **ln** is interrupted before ending, the destination path may have been removed without the new link having been created.

When creating a symbolic link, **ln** does not check that the *source_file* named actually exists, or even that it is a valid pathname. If the file does not exist or if the *source_file* is not a valid pathname, it will fail when any process attempts to access that link.

See also:

cp, **mv**

lockfd

© 2005, QNX Software Systems

Lock a floppy drive to one capacity/media type (QNX)

Syntax:

lockfd [-l *mediatype* **| -u] [***drivename***]**

Options:

-l *mediatype*	("el") Lock the drive at the specified *mediatype*, where *mediatype* is one of **360** or **1.2** for 5 $\frac{1}{4}$ inch disk drives, or **720**, **1.4** or **2.8** for 3 $\frac{1}{2}$ inch drives.
-u	Unlock the drive.
drivename	The pathname of a floppy block special file to lock (default is **/dev/fd0**).

Description:

In normal operation, the **Fsys.floppy** driver will automatically detect the media size of the disk occupying the drive by reading the first block of the disk, where QNX writes that information. If a disk contains just raw data (for instance, if it was a **tar** archive), the first block will contain the first 512 bytes of data instead and will not contain the drive information. Having an invalid or incorrect value will cause the driver to behave erratically. The **lockfd** utility can be used to tell the **Fsys.floppy** driver to lock the drive to a specific media size, and to ignore (and not attempt to read) this data from the first block of the disk. This ensures correct operation regardless of the data stored on the disk, at the expense of having to run **lockfd** each time the media size needs to be changed.

Files:

If an error occurs, the **lockfd** utility will write a diagnostic message to the standard error. The standard input and standard output are unused.

The utility acts on a single block special file; either the one specified on the command line as the *drivename* parameter, or **/dev/fd0** (the default). An *ioctl()* is sent to the device; currently the only driver which supports this *ioctl()* is the **Fsys.floppy** driver.

Exit status:

>0	An error occurred.
0	The drive was successfully locked to the specified media size.

See also:

Fsys, **Fsys.floppy**, **pax**, **tar**, **vol**

logger

© 2005, QNX Software Systems

Make entries in the system log (POSIX)

Syntax:

logger [-is] [-f *file***] [-p** *pri***] [-t** *tag***] [***message ...***]**

Options:

-i	Log the process id of the logger process with each line.
-s	Log the message to standard error, as well as the system log.
-f *file*	Log the contents of the specified *file*.
-p *pri*	Enter the message with the specified *priority*. The priority may be specified numerically or as a *facility***.***level* pair. For example, **-p local3.info** logs the message(s) as **informational** level in the **local3** facility. The default is **user.notice**.
-t *tag*	Mark every line in the log with the specified *tag*.
message	Write the *message* to log; if not specified, and the **-f** flag is not provided, standard input is logged.

Description:

The **logger** command provides a shell command interface to the **syslogd** daemon.

Examples:

Log the message “System rebooted”:

```
logger System rebooted
```

Log the contents of the file **/tmp/log** tagging each line with **log**:

```
logger -f /tmp/log -t log
```

Environment variables:

SYSLOG When defined, **SYSLOG** specifies which node **`syslogd`** is running on. By default, the local node is assumed.

Exit status:

0 Successful completion.

>0 An error occurred.

Because the syslog API does not return error codes, only argument errors can be detected.

See also:

syslog()

`syslogd`

login

© 2005, QNX Software Systems

Log in (UNIX)

Syntax:

login [-f *username***] [-p] [-t** *timeout***] [***username***]**

Options:

-f *username*	When run as **root** only, force login of *username*.
-p	Preserve current environment variables. If **-p** isn't specified, the current environment will be cleared and the only environment variables set will be **HOME**, **LOGNAME**, **SHELL**, and those in the **/etc/default/login** file (these will be set regardless of whether **-p** is specified).
-t *timeout*	Cancel the login if no response is received within the specified number of seconds. A value of 0 (the default) disables the timeout facility.
username	A user's login name.

Description:

The **login** command starts a session associated with a user. It is used when someone either first signs on to the computer, or wishes to log in as another user. If **login** is invoked without an argument, it prompts for a username and password. If invoked with an argument, the argument is assumed to be the username.

The **login** command verifies the username and password against a password database and, if verified, starts the user session. If either the username or password is unacceptable, **login** will ask again for the username and password. After five unsuccessful attempts it will print the message "**Login incorrect**" and terminate. If the file **/etc/nologin** exists, users other than **root** will be prevented from logging in. When a user attempts to log in, the contents of the **/etc/nologin** file will be written to the user's terminal, and **login** will exit.

The **login** utility sets the user ID and group ID as well as the current working directory, then executes a shell according to specifications found in the password database.

If execution of the shell fails, **login** prints the message "**No Shell**" and exits. The shell is started with a dash (**-**) prepended to its name as argument 0. This informs the shell to perform its startup routine.

The **login** utility sets the **HOME**, **SHELL**, and **LOGNAME** environment variables. The **login** utility also updates system accounting information.

The **login** utility will also set the environment variables specified in the **/etc/default/login** file. This file lets you specify which environment variable settings are to be used across login sessions. If a variable isn't set when **login** is started and this file contains a default for the variable, that default will be used. If the file doesn't contain a default for the variable, the variable will be cleared by **login**. If a variable is already set when **login** is started and **-p** is specified, it is preserved, even if the **/etc/default/login** file contains a default for the variable. The **-p** option allows site-specific environment variables to be passed from the system startup processes (e.g. **modem**, **tinit**) down to applications.

The **login** utility may be executed only from the login shell, or when no session is active. You can't nest **login** commands.

Since **login** is a shell alias for **exec login**, the following command won't work as expected:

on -t *device* **login**

You can use the following command instead:

on -t *device* **/bin/login**

© 2005, QNX Software Systems

Files:

/etc/default/login

Sets login environment variables.

/etc/passwd

Contains the user database (username, user ID, group ID, full name, home directory, shell).

/etc/shadow

Contains encrypted login password for each userid.

/etc/acclog

Accounting records are appended to this file.

See also:

logout, **sh**

logname

Return user's login name (POSIX)

Syntax:

logname

Options:

None.

Description:

The **logname** utility writes the user's login name to standard output. If the user doesn't have an entry in the password file, a diagnostic message is written to standard error and **logname** exits with an exit status greater than zero.

Files:

/etc/passwd

Password file; contains valid userids on the system.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

id, **who**

logout

© 2005, QNX Software Systems

Terminate a session (UNIX)

Syntax:

logout

Options:

None.

Description:

The **logout** command terminates a session. This command can be executed only from a shell invoked by a **login** command; otherwise a message is printed and the **logout** command is ignored.

If you execute a **login** command from the **login** shell, the shell from which it is run will be terminated. This is because by default **login** is an alias for **exec login**.

See also:

login

Syntax:

lp [-cv] [-d *name***] [-P** *name***] [-n** *copies***] [***file***...]**

Options:

-c Exit only after further access to any of the input files is no longer required. Note that the QNX implementation of **lp** exits this way, whether the **-c** option is specified or not.

-d *name* Use the named spooler and/or queue. If *name* begins with a slash, it is assumed to be a full name, where the last component is the name of a spool queue and the prefix is the name of the spooler (*/spooler/queue*). If *name* does *not* begin with a slash, it is assumed to be a queue name.

If a server is not specified, the utility will use the spooler identified in the **LPSRVR** environment variable, or **/qnx/spooler** if **LPSRVR** is not set.

If a queue is not specified, **lp** will use the queue identified in the **LPDEST** environment variable, or the first queue defined in the setup file of the spooler being used if **LPDEST** is not set.

For more detailed discussion, see Ch. 9, "Print Spooling," in the QNX *Installation & Configuration* manual.

-n *copies* Write this number of copies of the files, where *copies* is a positive decimal integer. Note that this does not enqueue two copies, but rather sets **$(ncopies)** metavariable which may be utilized by a copy-in or copy-out rule. See the QNX *Installation & Configuration* manual for a discussion on print spooling.

-P *name* (QNX extension) Same functionality as the **-d** *name* option.

-v	(QNX extension) Be verbose; print information about submitted jobs.
file	The pathname of a file to be printed. If no files are specified, **lp** uses the standard input.

Description:

The **lp** utility enqueues a file into a print spooler queue. The default printer is used, unless one is specified with the **-P** *name* option. The actual writing of files to the output device may occur after **lp** successfully exits.

Examples:

Print the file **myfile**:

```
lp myfile
```

Print multiple files with headers:

```
pr file1 file2 | lp
```

Environment variables:

LPSRVR	Use this spool server by default. (Will use **/qnx/spooler** if not set.)
LPDEST	Use this spool queue by default. (Will use first queue entry in the setup file of the located spooler if not set.)
PRINTER	Historical alternative to **LPDEST**. Ignored if **LPDEST** is set.

Exit status:

0 All input files were processed and output successfully.

>0 No output device was available, or an error occurred.

See also:

lpc, **lpq**, **lprm**, **lpsrvr**, **pr**

Installation & Configuration

lpc

© 2005, QNX Software Systems

Line printer control program (UNIX)

Syntax:

lpc [*command* **[***argument***...]]**

Options:

See description.

Description:

The **lpc** utility is a tool to control the print services. There are two sides to the printing process: the *enqueuing* of jobs and the *dequeuing* of jobs. Jobs are enqueued by **lp** and dequeued by the print server (**lpsrvr**). The **lpc** utility communicates with the print server to control such things as allowing new jobs to be enqueued, stopping the dequeueing of jobs, reordering jobs in the queue, etc.

The print spooler used is that defined by the **LPSRVR** environment variable. If **LPSRVR** is not set, **/qnx/spooler** is used.

Control commands are specified as additional arguments on the **lpc** command line. When no commands are specified, they will instead be read from standard input.

Brief online help is available through the **help** subcommand:

lpc help [*command...***]**

The help command gives a short description of each command specified. If no command is specified, a list of the available commands is displayed.

Commands

enable {all|*printer...***}**

Allow enqueuing of jobs on the specified printers.

disable {all|*printer...***}**

Prevent enqueuing of jobs on the specified printers.

start {all|*printer*...**}**

Allow enqueuing and dequeuing of jobs on the specified printers.

stop {all|*printer*...**}**

Prevent enqueuing and dequeuing of jobs on the specified printers.

status {all|*printer*...**}**

Display the status of jobs on the specified printers.

abort {all|*printer*...**}**

Abort the current print job and disable the dequeuing process on the specified printers.

clean {all|*printer*...**}**

Remove all enqueued print jobs from the specified printers.

restart {all|*printer*...**}**

Examine the status of the dequeuing process, and, if necessary, restart the process on the specified printers.

topq *printer* **{***jobnum*...**|***user*...**}**

Change the order of jobs for the specified printer. The order is left to right and is specified either by job number or user name. Specifying by user name means all jobs owned by that user.

move *fromprinter toprinter* **{all|***jobnum*...**|***user*...**}**

Move the specified jobs from *fromprinter* to *toprinter*. Move the "current job"† if no job is specified.

pause {all|*printer*...**}**

Stop the dequeuing process on the specified printers. Allow the "current job"† to complete.

resume {all|*printer*...**}**

Start the dequeuing process on the specified printers.

END-OF-FILE *or* **exit** *or* **quit**

Exit from **lpc**.

† The “current job” refers to the job currently being printed.

Environment variables:

LPSRVR Use this spool server. (Will use **/qnx/spooler** if not set.)

See also:

lp, **lpq**, **lprm**, **lpsrvr**

TCP/IP **lpr*** utilities

Installation & Configuration

lpq

Display print queue status (UNIX)

Syntax:

lpq [-l] [-P *name***] [-q] [-s** *seconds***] [***jobnumber***...] [***user***...]**

Options:

-l ("el") Loop. After displaying the queue, sleep for a period of time (default is 5 seconds), then redisplay the queue.

-P *name* Use the named spooler and/or queue. If *name* begins with a slash, it is assumed to be a full name, where the last component is the name of a spool queue and the prefix is the name of the spooler (*/spooler/queue*). If *name* does *not* begin with a slash, it is assumed to be a queue name.

If a server is not specified, the utility will use the spooler identified in the **LPSRVR** environment variable, or **/qnx/spooler** if **LPSRVR** is not set.

If a queue is not specified, **lp** will use the queue identified in the **LPDEST** environment variable, or the first queue defined in the setup file of the spooler being used if **LPDEST** is not set.

For more detailed discussion, see Ch.9, "Print Spooling," in the QNX *Installation & Configuration* manual.

-q Print a list of the available printers.

-s *seconds* When looping, sleep for *seconds* rather than the default. Setting this value automatically sets looping mode.

jobnumber
user Specifying either of these causes **lpq** to filter out only those jobs of interest.

Description:

The **lpq** utility reports on the status of jobs in a printer queue.

For each job, **lpq** reports the user's name, the job identifier, the ranking in the queue, and the names of files that make up the job.

If no filename is available, the file name will be displayed as "standard input".

When **lpq** is set for "looping," it will continue to redisplay the entries until no entries are left to display.

Examples:

Display all current print jobs submitted by **steve**:

```
lpq steve
```

Environment variables:

LPSRVR	Use this spool server by default. (Will use **/qnx/spooler** if not set.)
LPDEST	Use this spool queue by default. (Will use first queue entry in the setup file of the located spooler if not set.)
PRINTER	Historical alternative to **LPDEST**. Ignored if **LPDEST** is set.

Exit status:

0	Jobs of interest were discovered.
1	No jobs of interest were discovered.
>1	An error occurred.

Caveats:

Since **lpq** is based on polling, it's possible that very fast jobs will completely escape its notice.

See also:

lp, **lpc**, **lprm**, **lpsrvr**

Installation & Configuration

lprm

© 2005, QNX Software Systems

Remove jobs from the printer queue (UNIX)

Syntax:

lprm [-P *name***] [-a] [***jobnumber***...] [***user***...]**

Options:

-a Remove all jobs owned by the user. If the user is **root**, then the spool queue will be emptied entirely.

-P *name* Use the named spooler and/or queue. If *name* begins with a slash, it is assumed to be a full name, where the last component is the name of a spool queue and the prefix is the name of the spooler (*/spooler/queue*). If *name* does *not* begin with a slash, it is assumed to be a queue name.

If a server is not specified, the utility will use the spooler identified in the **LPSRVR** environment variable, or **/qnx/spooler** if **LPSRVR** is not set.

If a queue is not specified, **lp** will use the queue identified in the **LPDEST** environment variable, or the first queue defined in the setup file of the spooler being used if **LPDEST** is not set.

For more detailed discussion, see Ch. 9, "Print Spooling" in the QNX *Installation & Configuration* manual.

jobnumber
user Specifying either of these causes **lprm** to filter out only those jobs of interest (i.e. either a specific job, or all the jobs owned by the specified particular user).

Description:

The **lprm** utility removes a job or jobs from a printer spool queue. If **lprm** is invoked with no arguments, it will abandon the currently active job, provided the user invoking the **lprm** utility owns the job or has appropriate privileges.

Specifying user names is useful only to the superuser (**root**). All other users can remove only their own jobs.

Environment variables:

LPSRVR Use this spool server by default. (Will use **/qnx/spooler** if not set.)

LPDEST Use this spool queue by default. (Will use first queue entry in the setup file of the located spooler if not set.)

PRINTER Historical alternative to **LPDEST**. Ignored if **LPDEST** is set.

Caveats:

With the increasing amount of memory built into newer printers, pending jobs may be physically stored within the printer itself, and **lprm** may not be able to remove them. Once the job has been sent to the printer, it is too late to stop it using **lprm**.

See also:

lp, **lpc**, **lpq**, **lpsrvr**

TCP/IP **lpr*** utilities

Installation & Configuration

lpsrvr

© 2005, QNX Software Systems

Print server (QNX)

Syntax:

lpsrvr [-c *command***] [-f** *setup_file***]**
[-n *children***] [-Ssv] &**

Options:

-c *command*	Use this as the default spooling command.
-f *setup_file*	Load this setup file. (Default: **/etc/config/lpsrvr.***node*; if not found, then **/etc/config/lpsrvr** is used instead.)
-n *children*	Limit the number of concurrent child processes to this number. (Default: 64)
-s	Spawn commands. Don't use the Shell.
-S	Share. Allow multiple spoolers to share a common working directory.
-v	Be verbose.

Description:

The **lpsrvr** manager is the print server. It accepts requests for printing, rescheduling, and examining of print jobs. When **lpsrvr** starts, it scans the configuration file to find out about existing printers, then scans for any residual work from a previous invocation.

☞ You must be logged in as **root** to run **lpsrvr**.

Files:

/etc/config/lpsrvr

Configuration file.

See also:

lp, **lpc**, **lpq**, **lprm**

Installation & Configuration

ls

© 2005, QNX Software Systems

List directory contents (POSIX)

Syntax:

ls [-acCdFilqrRstu1] [-bDfghnopSvx] [*file***...]**

Options:

-1 ("one") Force output to be one entry per line.

-a List all files, including those which start with a dot (**.**). By default, these entries aren't listed.

-b (QNX extension) Use the size of the file for sorting and printing. Sort in descending order.

-c For sorting (**-t**) or printing (**-l** ("el")), use time of last change to the file's status information instead of time of last modification of the file itself.

-C Display multiple-column output, with entries sorted down the columns according to the collating sequence.

-d Treat directories like files — give information on the directory itself, not on the files or subdirectories it contains.

-D (QNX extension) Display directories only.

-f (QNX extension) Don't sort the output (same as **-S**).

-F Indicate the filetype by adding an extra character after some pathnames, as follows:

Character	Meaning
/	for directories
*****	for executable files
|	for FIFOs (named pipes)
@	for named special files (QNX extension)

-g (QNX extension) List in long format, as in **ls -l** ("el"), but don't show owner (group is displayed).

-h (QNX extension) Display a header for the **-l** ("el") **-x** and **-n** options.

-i For each file, print the file's serial number (inode).

-l ("el") List in long format. This option provides most of the relevant file information such as permissions, link count, owner/group of the file, as well as the size, date, and name of the file, as follows:

```
drwxrwxrwx  7 root     techies   22528 Jan 17 15:38 Csrc
-rw-rw-rw-  1 barney   rubble       22 Feb 14 13:41 rock
-rwxrwxrwx  1 root     techies      22 Feb 14 13:41 exec
-rw-rw-rw-  1 fred     fstone       22 Feb 14 13:41 fred
drwxrwxrwx  2 root     techies   23040 Feb 12 10:56 libt
drwxrwxrwx  2 root     root       2048 Sep 28 06:39 util
```

Note that the first character preceding the permissions denotes the file type, as follows:

Character	File type
-	Regular
b	Block special
c	Character special
d	Directory
p	FIFO
l	Symbolic link
n	Named special
s	Socket special

-n (QNX extension) Same as **-l** ("el"), except display group ID and user ID numbers instead of names.

-o (QNX extension) List in long format, as in **ls -l** (“el”), but don’t show group (owner is displayed).

-p (QNX extension) Display non-directory files only, as a list of pathnames relative to the original *file* specified on the command line, or the current directory (**.**) if no *file* parameters were supplied. This option allows you to pass full pathnames of files to programs (i.e. the portion of the path supplied on the command line will not be stripped off from the filenames in **ls**’s output).

-q Force filename characters that aren’t included in the character set classification in the current locale to be displayed as a question mark (**?**). This is the default if output is to a terminal.

-r Reverse the order of the sort. When sorting alphabetically (default), this will result in reverse alphabetical order. When sorting by time (**-t**), files will be listed oldest to newest. When sorting by size (**-b**), files will be listed smallest to largest.

-R Recursively list all subdirectories encountered.

-s Display the size of the file in 512-byte blocks.

-S (QNX extension) Don’t sort the output. This option is useful for determining the order in which entries are found in a given directory.

-t Sort by time modified (most recently modified first) before sorting the files by the collating sequence.

-u For sorting (**-t**) or printing (**-l** (“el”)), use time of last access (i.e. last use), instead of time of last modification of the file.

-v (QNX extension) List directories first.

-x (QNX extension) Display extent information for a file or directory. Extent information describes how a file is segmented on disk. This option displays information similar to the following example:

```
 Size xtnts xblk_1  xsize   xblk status Filename
22528     8 000686  4     000C1A  ----  Csrc
   22     1 00adc3  1     000000  --I-  rock
   22     1 00ACD8  1     000000  ----  exec
   22     1 00ADC2  1     000000  -B--  results
23040    10 004A55  4     005A91  ----  libtests
 2048     1 0013EB  8     000000  ---G  logfile
```

Note that the text descriptions of the columns are displayed only if the **-h** option is used. The meaning of the fields is as follows:

Size	File size, in bytes.
xtnts	Number of extents in the file.
xblk_1	The block address (hex) of the first extent in the file.
xsize	The number of blocks in the first extent of the file.
xblk	The block address (hex) of the first extent block for the file. If there are no extent blocks (i.e. the file only has zero or one extent), this will be zero.
status	A subset of the status bits related to the file. If a bit is set, the corresponding letter will appear. If not set, a dash (**-**) will appear instead. The bits shown are:

Bit:	Represents:	Meaning:
B	_FILE_BUSY	Indicates that the file has been grown or truncated but the inode has not been updated on disk.
I	_FILE_INODE	The directory entry does not contain an embedded inode. Inode information for this file is stored in the **.inodes** file.

continued...

© 2005, QNX Software Systems

Bit:	Represents:	Meaning:
C	_FSYS_CLEAN	Valid for the filesystem root only; indicates that there are no busy files.
G	_FILE_GROWN	Pregrown file. Indicates that file space has been preallocated in excess of the space currently used by the file in anticipation of future growth.

Filename — The name of the file. If **-F** is supplied, and the filename will have the file type character appended (e.g. **/** for directories, ***** for executables etc.).

file — The pathname of a file to be listed. If the file specified isn't found, a diagnostic message is output on standard error.

Description:

For each file you name that isn't a directory, **ls** displays the file's name as well as any information requested on the file.

For each directory you name, **ls** displays the names of files contained within that directory, as well as any information requested on the files. The **-d** option overrides this behavior and makes **ls** display information on the directory itself, rather than on its contents.

If you specify more than one file, **ls** displays files that aren't directories first. Directories and non-directories are sorted separately.

If you don't specify a file, **ls** displays the contents of the current directory.

Specifying more than one of the **-C**, **-l** ("el"), and **-1** ("one") options isn't considered an error. The last option specified determines the output format.

In many environments, the **ls** command will be "aliased" to either **ls -C** or **ls -CF**, the two most common **ls** display formats. Unless the

POSIX_STRICT environment variable is set, **ls** defaults to the multi-column output (**-C** option) when its standard output is a tty. When the standard output is not a tty, it defaults to single-column output.

The **-p** option is useful for passing a list of all non-directory filenames, one filename per line, to other programs. The filenames include the full pathnames.

When displaying a timestamp for a file, **ls** will display the date and time, unless the file is older or newer than the current date by six months (a "month" is defined as 30 days). Otherwise, **ls** will display the date and year.

Environment variables:

COLUMNS If this variable contains a string representing a decimal integer, it indicates the user's preferred column position width for displaying multiple-column output. The **ls** utility calculates how many pathname text columns to display (see **-C**) based on the width provided. If **COLUMNS** isn't set or is invalid, the number of columns displayed is determined by the type of output device.

POSIX_STRICT

Interpret options according to POSIX specifications.

TZ Determines the time zone for date and time displays.

Exit status:

0 All files were listed successfully.

>0 An error occurred.

© 2005, QNX Software Systems

See also:

find

Name:

/etc/magic

Description:

The **file** command identifies the type of a file using, among other tests, a test for whether the file begins with a certain *magic number*. The file **/etc/magic** specifies what magic numbers are to be tested for, what message to print if a particular magic number is found, and additional information to extract from the file.

Each line of the file specifies a test to be performed. A test compares the data starting at a particular offset in the file with a 1-byte, 2-byte, or 4-byte numeric value or a string. If the test succeeds, a message is printed. The line consists of the following fields:

offset A number specifying the offset, in bytes, into the file of the data which is to be tested.

type The type of the data to be tested. The possible values are:

byte A one-byte value.

short A two-byte value (on most systems) in this machine's native byte order.

long A four-byte value (on most systems) in this machine's native byte order.

string A string of bytes.

date A four-byte value interpreted as a UNIX date.

beshort A two-byte value (on most systems) in big-endian byte order.

belong A four-byte value (on most systems) in big-endian byte order.

bedate A four-byte value (on most systems) in big-endian byte order, interpreted as a UNIX date.

© 2005, QNX Software Systems

leshort A two-byte value (on most systems) in little-endian byte order.

lelong A four-byte value (on most systems) in little-endian byte order.

ledate A four-byte value (on most systems) in little-endian byte order, interpreted as a UNIX date.

The numeric types may optionally be followed by **&** and a numeric value, to specify that the value is to be ANDed with the numeric value before any comparisons are done. Prepending a **u** to the type indicates that ordered comparisons should be unsigned.

test The value to be compared with the value from the file. If the *type* is numeric, this value is specified in C form; if it's a *string*, it's specified as a C string with the usual escapes permitted (e.g. **\n** for newline).

Numeric values may be preceded by a character indicating the operation to be performed:

- **=**, to specify that the value from the file must equal the specified value
- **<**, to specify that the value from the file must be less than the specified value
- **>**, to specify that the value from the file must be greater than the specified value
- **&**, to specify that the value from the file must have set all of the bits that are set in the specified value
- **^**, to specify that the value from the file must have clear any of the bits that are set in the specified value
- **x**, to specify that any value matches.

If the character is omitted, it's assumed to be **=**.

Numeric values are specified in C form; e.g. **13** is decimal, **013** is octal, and **0x13** is hexadecimal.

For string values, the byte string from the file must match the specified byte string. The operators **=**, **<** and **>** (but not **&**) can be applied to strings. The length used for matching is that of the string argument in the **/etc/magic** file. This means that a line can match any string, and then presumably print that string, by doing **>\0** (because all strings are greater than the null string).

message The message to be printed if the comparison succeeds. If the string contains a *printf()* format specification, the value from the file (with any specified masking performed) is printed using the *message* as the format string.

Some file formats contain additional information that's to be printed along with the file type. A line that begins with the character **>** indicates additional tests and messages to be printed. The number of **>** characters on the line indicates the level of the test; a line with no **>** at the beginning is considered to be at level 0. Each line at level *n+1* is under the control of the line at level *n* most closely preceding it in the magic file.

If the test on a line at level *n* succeeds, the tests specified in all the subsequent lines at level *n+1* are performed, and the messages printed if the tests succeed. The next line at level *n* terminates this.

If the first character following the last **>** is a **(**, then the string after the parenthesis is interpreted as an indirect offset. That means that the number after the parenthesis is used as an offset in the file. The value at that offset is read, and is used again as an offset in the file. Indirect offsets are of the form: **((***x***[.[bsl]][+-][***y***])**. The value of *x* is used as an offset in the file. A byte, short or long is read at that offset depending on the **[bsl]** type specifier. To that number the value of *y* is added and the result is used as an offset in the file. The default type if one isn't specified is long.

© 2005, QNX Software Systems

Caveats:

The formats **long**, **belong**, **lelong**, **short**, **beshort**, **leshort**, **date**, **bedate**, and **ledate** are system-dependent; perhaps they should be specified as a number of bytes (2B, 4B, etc), since the files being recognized typically come from a system on which the lengths are invariant.

There's (currently) no support for specified-endian data to be used in indirect offsets.

See also:

file

Syntax:

mailx [-s *subject***]** *address***...**

mailx [-eFHinN] [-f [*file***]] [-u** *user***]**

Options:

-e	Test for the presence of mail. If there is mail to read, **mailx** prints nothing and exits with a successful return code.
-f [*file*]	Read messages from the named file instead of from the mailbox. If no file is specified, the *mbox* is used, which defaults to **mbox** in your home directory.
-F	Record the message in a file named after the first recipient. This option overrides the *record* variable.
-H	Display a header summary only.
-i	Ignore interrupts.
-n	Don't initialize from the system default **mailx.rc** file.
-N	Don't display an initial header summary.
-s *subject*	Set the Subject header field to *subject*.
-u *user*	Read the named user's mailbox. Without appropriate privileges, you can read only your own mailbox.
address	The address passed to the mail delivery software.

Description:

The **mailx** utility is an interactive program that lets you receive, compose, and send messages. When you read mail, **mailx** provides commands that let you browse, save, delete, and answer messages.

© 2005, QNX Software Systems

When you send mail, **mailx** lets you interactively review and edit your messages; you can also include text from files and other messages.

Starting mail

When you start **mailx**, it reads commands from the **/usr/lib/mailx.rc** file to initialize certain parameters. It then reads from a private startup file to get your personal commands and variable settings. This private startup file is normally called **.mailrc** and resides in your home directory (you can specify another pathname with the **MAILRC** environment variable).

You can place most **mailx** commands in your startup file (the ones most commonly used are those to set up initial display options and alias lists). Note, however, that you *can't* use the following commands:

!	**followup**	**Reply**
Copy	**hold**	**reply**
edit	**mail**	**shell**
Followup	**preserve**	**visual**

If an error is found in the startup file, all the remaining lines are ignored.

Reading mail

If you specify no recipients when you invoke **mailx**, it enters *command mode*. If you have no unread messages, **mailx** will print the following message and then exit:

No mail for *username*

If mail is waiting, however, you'll first see a header summary of the first several messages, followed by a prompt indicating that you can enter regular commands (described below). By default, the prompt is **&**, but you can choose another one with the *prompt* variable.

Each message is assigned a number. At any time, there is a *current message*, which is marked by a **>** in the header summary. To read this message, you can use the **print** command (which you can abbreviate to a simple **p**).

Many commands, including the **print** command, let you specify an optional list of messages (*msglist*) to work on. If you omit the *msglist*, the command applies to the current message. A *msglist* is a list of message specifications, separated by spaces, which can include:

Message Specification	Meaning
n	message number *n*
.	the current message
^	the first undeleted message, or the first deleted message for the **undelete** command
$	the last message
*****	all messages
n-m	an inclusive range of message numbers
user	all messages from *user*
/*string*	all messages with *string* in the subject line (case ignored)
:*c*	all messages of type *c*

Valid types of messages for **:***type* message specs are:

d deleted messages

n new messages

o old messages

r read messages

u unread messages

The context of the command determines whether this type of message specification makes sense.

Additional arguments are usually strings whose usage depends on the command involved. Filenames, where expected, can be specified with metacharacters understood by the Shell. Some commands recognize special characters; these are documented with the commands below.

Incoming mail is stored in a separate file for each user, called a *mailbox*. When you use **mailx** to read messages, it looks in your mailbox by default. As you read messages, they are marked to be moved to a secondary file for storage, unless you take action otherwise. The secondary file, called the *mbox*, is normally located in your home directory. Messages remain in this file until you explicitly remove them.

Sending mail

To send mail, you can invoke **mailx** with the user names of the people you wish to send to — this puts **mailx** in *input mode* and you can begin typing in your message. If you didn't specify any subject via option **-s** when invoking **mailx**, you'll be prompted for a subject, provided that the *asksub* variable is set. Once you've entered your subject, **mailx** enters input mode and you can begin typing.

As you type your message, **mailx** stores it in a temporary file. To review or modify the message, you can use the tilde (~) escape commands described below. To submit your message, you enter either of the following on a line by itself:

```
~.        or    .
```

When you indicate your message is complete, your message is routed to each recipient.

Common commands

The "Commands" and "Tilde escapes" sections below describe all the commands available in **mailx**. If you're a new user, the following table lists the basic commands you'll use most often:

In order to:	Use:
display a message	**print**
reply to a message	**reply** *or* **respond**
forward a message (input mode)	**~f**
delete a message	**delete**
restore a message deleted in the current session	**undelete** *or* **exit**
end a mail session	**quit**
create distribution lists so you can send mail to groups of people	**alias**
display a summary of commands	**?**
display a summary of tilde escape commands (input mode)	**~?**

Modifying mailx's behavior

You can modify how **mailx** behaves through a set of variables. These are flags and valued parameters that you set and clear via the **set** and **unset** commands. These variables are described under "Internal variables," below.

Commands

!*command* Escape to the shell.

*comment* Null command (comment).

= Display the current message number.

? Display a summary of commands.

a[**lias**] *alias name...*

Declare an alias for the given list of recipients. The names will be substituted when *alias* is named as a recipient.

alt[**ernates**] *name...*

Declare a list of alternate names for your login. When responding to a message, these names are removed from the list of recipients for the response. With no arguments, **alternates** displays the current list of alternate names.

cd [*directory*]
ch[**dir**] [*directory*]

Change directory. If a directory isn't specified, your home directory is used.

C[**opy**] [*msglist*]

Save the specified messages in a file whose name is derived from the author of the message. The messages aren't marked as saved. Otherwise equivalent to the **Save** command.

c[**opy**] [*file*]
c[**opy**] [*msglist*] *file*

Copy messages to the named file without marking the messages as saved. If a message list isn't specified, the current message is used. Otherwise equivalent to the **save** command.

d[**elete**] [*msglist*]

Delete messages from *mailbox*. If *autoprint* is set, print the message following the last deleted message.

di[**scard**] [*header-field*...]

When displaying messages, don't print the specified header fields (e.g. *Message Id*, *Received*, ...). The fields are included when the message is saved. The **Print** and **Type** commands override this command.

dp [*msglist*]
dt [*msglist*]

Delete the specified messages from the *mailbox* and print the message following the last deleted message. This is roughly equivalent to a **delete** command followed by a **print** command.

ec[ho] *string...*

Echo the given strings.

e[**dit**] [*msglist*]

Edit the given messages. The messages are placed in a temporary file and the *EDITOR* variable is used to get the name of the editor. (Default: **vi**)

ex[**it**]

Exit from **mailx** without changing the mailbox. No messages are saved in the *mbox* (see also **quit**).

fi[**le**] [*file*]
fold[**er**] [*file*] Quit from the current file of messages and read in the named mailbox file. Instead of specifying a filename, you can also use one of the following:

%	the current mailbox
%*user*	the mailbox for *user*
#	the previous file
&	the current mbox
+*file*	the named file in the folder directory (see the *folder* variable)

All of these special forms are recognized when used for any file arguments for **mailx** commands, tilde escapes, or the **-F** option. The default file is the current mailbox.

folders Display the names of the files in the directory set by the *folder* variable.

F[**ollowup**] [*msglist*]

Respond to the first message in *msglist*, sending the message to the author of each message in the *msglist*. The subject line is taken from the first message and the response is recorded in a file whose name is derived from the author of the first message. See also the **Copy**, **followup**, and **Save** commands as well as the *outfolder* variable.

fo[**llowup**] [*message*]

Respond to a message, recording the message in a file whose name is derived from the author of the message. The subject line is taken from the subject of the first message. See also the **Copy**, **Followup**, and **Save** commands as well as the *outfolder* variable.

f[**rom**] [*msglist*]

Display the header summary for the specified messages (default is the current message).

g[**roup**] *aliasname*...

Same as the **alias** command.

h[**eaders**] [*message*]

Display the page of headers that includes the specified message. The *screen* variable sets the number of headers per page. See also the **z** command.

hel[**p**] Display a summary of commands.

ho[**ld**] [*msglist*]

Hold the specified messages in the mailbox.

i[**f**] **s|r**
mail-commands
el[**se**] *mail-commands*
en[**dif**]

Conditional execution, where: **s** executes the following *mail-command*s, up to an **else** or **endif**, if the program is in *send* mode, and **r** causes *mail-commands* to be executed only in *receive* mode.

ig[**nore**] *header-field*...

Same as the **discard** command.

l[**ist**] Display all commands available. No explanations are given.

m[**ail**] *name*... Mail a message to the named users.

mb[**ox**] [*msglist*]

Arrange for the given messages to end up in the standard *mbox* save file when **mailx** terminates

normally. For a description of this file, see the *MBOX* variable. See also the **exit** and **quit** commands.

n[**ext**] [**message**]

Go to the next message that matches *message*. You could specify a message list, but note that in this case the first valid message in the list is the only one used. See the discussion of *msglist*s above for a description of possible message specifications.

pi[**pe**] [*msglist*] [*command*]
|[*msglist*] *command*

Pipe the message through the given *command*. The command can be given options. The message is treated as if it were read. If no arguments are given, the current message is piped through the command specified by the *cmd* variable. If the *page* variable is set, a **form-feed** character is inserted after each message.

pre[**serve**] [*msglist*]

Same as the **hold** command.

P[**rint**] [*msglist*]

Display the specified messages. All header lines are displayed, including any otherwise suppressed by the **ignore** and **discard** commands.

p[**rint**] [**msglist**]

Display the specified messages. If the *crt* variable is set, any messages longer that the number of lines specified by *crt* are paged through the command specified by the *PAGER* variable.

q[**uit**]

Exit from **mailx**, and do the following:

- store messages that were read in mbox (unless *hold* is set)

- delete messages that have been explicitly saved (unless *keepsave* is set)
- discard messages that have been deleted
- save all remaining messages in the mailbox

R[**eply**] [*msglist*] **R**[**espond**] [*msglist*]

Send a response to the author of each message in the *msglist*. The subject line is taken from the first message. If *record* is set to a filename, the response is appended to that file.

r[**eply**] [*message*]
r[**espond**] [*message*]

Reply to the specified message, including all other recipients of the message. The subject line is taken from the subject of the first message. If **record** is set to a filename, the response is appended to that file.

S[**ave**] [*msglist*]

Save the specified messages in a file whose name is derived from the author of the first message. The name of the file is taken to be the author's name. See also the **Copy**, **Followup**, and **followup** commands as well as the *outfolder* variable.

s[**ave**] [*file*]
s[**ave**] [*msglist*] *file*

Save the specified messages in the named file. If the file doesn't exist, it will be created. The message is deleted from the *mailbox* when **mailx** terminates unless the *keepsave* variable is set. See also the **exit** and **quit** commands.

s[**et**]
se[**t**] *name*
se[**t**] *name*=*string*
se[**t**] **name=***number*
se[**t**] **no**name

Define a variable called *name*. You can give the variable a null, string, or numeric value. Quoting and backslash escapes can occur anywhere in *string*. If you use **set** by itself, it displays all defined variables and their values. See the section on "Internal variables" for a description of all predefined variables.

sh[**ell**] Invoke the Shell. See also the *SHELL* variable.

si[**ze**] [*msglist*]

Display in bytes the size of the specified messages.

so[**urce**] *file* Read commands from the named file and return to command mode.

to[**p**] [*msglist*] Display the top few lines of the specified message (default is 5). You can use the *toplines* variable to determine the number of lines to be displayed.

tou[**ch**] [*msglist*]

Touch the specified messages. If any message in *msglist* isn't specifically saved in a file, it is placed in the *mbox* upon normal termination. See also the **exit** and **quit** commands.

T[**ype**] [*msglist*]

Same as the **Print** command.

t[**ype**] [*msglist*]

Same as the **print** command.

u[**ndelete**] [*msglist*]

Restore the specified deleted messages. This command restores only those messages deleted in

the current mail session. If the *autoprint* variable is set, the last message restored is displayed.

uns[**et**] *variable...*

Erase the specified variables. You cannot erase a variable that was imported from the execution environment (i.e. an environment variable).

ve[**rsion**] Display the current version and release date of **mailx**.

v[**isual**] [*msglist*]

Edit the given messages with a screen editor. The messages are placed in a temporary file and the *VISUAL* variable is used to get the name of the editor.

w[**rite**] [*msglist*] *file*

Write the given messages to the named file, minus the header lines and the following empty line. Otherwise equivalent to the **save** command.

x[**it**] Same as the **exit** command.

z[**+|-**] Scroll the header display forward or backward one screenful. The number of headers displayed is set by the *screen* variable.

Tilde escapes

You can enter the following commands only in input mode. To type a tilde character (**~**) as text, enter it twice. To change this special character, see the *escape* variable.

~! *command* Escape to the Shell.

~. Simulate end of file (terminate message input).

© 2005, QNX Software Systems

~: *mail-command*
~_ *mail-command*

Perform the specified command. This is valid only when you're sending a message while reading mail.

~? Display a summary of tilde escapes.

~A Insert the autograph string *Sign* into the message (see "Internal variables").

~a Insert the autograph string *sign* into the message (see "Internal variables").

~b *name...* Add the specified names to the blind carbon copy (Bcc) list.

~c *name...* Add the specified names to the carbon copy (Cc) list.

~d Read in the **dead.letter** file. For a description of this file, see the *DEAD* variable.

~e Invoke the editor to edit the partial message. The editor to be used is determined by the *EDITOR* variable.

~f [*msglist*] Forward the specified messages. The messages are inserted into the message, without alteration.

~h Prompt for Subject line and To, Cc, and Bcc lists. If the field is displayed with an initial value, you can edit it as if had just been typed.

~i *string* Insert the value of the named variable into the text of the message. For example, specifying **~iSign** is the same as specifying **~A**.

~m [*msglist*] Insert the specified messages into the letter, shifting the new text one tab stop to the right. This is valid only when you're sending a message while reading mail.

~p	Display the message being entered.
~q	Quit from the input mode by simulating an interrupt. If the body of the message isn't empty, the partial message is saved in the **dead.letter** file. For a description of this file, see the *DEAD* variable.
r *file* **~<**_file_	
~<!*command*	Read in the text of the named file or the standard output of the specified command.
~s *string*	Set the subject line to *string*.
~t *name...*	Add the given names to the To list.
~v	Invoke a visual editor to edit the partial message. The editor to be used is determined by the *VISUAL* variable.
~w *file*	Write the partial message to the named file, but don't include the header.
~x	Exit as with **~q**, but don't save the message in the **dead.letter** file.
~| *command*	Pipe the body of the message through the given command. If the command returns successfully, the output of the command replaces the message. Otherwise the message remains unchanged.

Internal variables

You can initialize the following variables within the **.mailrc** file; you can also set or modify them with the **set** command at any time. You can also import these variables from the execution environment, if they are already set. To erase variables, you can use the **unset** command. You can also erase variables using this command:

set no*variable*

© 2005, QNX Software Systems

which is equivalent to:

unset *variable*

Note that the following internal variables are initially assigned values that are imported from the execution environment. Nevertheless, those values can be overridden by assignments inside one of the startup files: *DEAD*, *EDITOR*, *MBOX*, *LISTER*, *PAGER*, *SHELL*, *VISUAL*.

append	Upon termination, append messages to the end of the mbox file instead of prepending them. The default is *noappend*.
askcc	Prompt for the Cc list after the message is entered. The default is *noaskcc*.
asksub	Prompt for the subject if it isn't specified on the command line with the **-s** option. Enabled by default.
autoprint	Enable automatic display of messages after **delete** and **undelete** commands. The default is *noautoprint*.
bang	Enable the special-case treatment of exclamation points (**!**) in escape command lines (as in **vi**). The default is *nobang*.
cmd=*command*	Set the default utility (including options) for the **pipe** command. There is no default value.
crt=*number*	Pipe messages having more than *number* lines through the command specified by the *PAGER* variable. Disabled by default.
DEAD=*file*	The pathname of a file in which to save partial letters in case of untimely interrupt or delivery errors. The default is **dead.letter** in your home directory.

dot	Take a period on a line by itself during input from a terminal as end-of-file. The default is *nodot*.
EDITOR=utility	The utility to run when the **edit** command or the **~e** command is used. Will default to the setting of the **EDITOR** environment variable. If the environment variable isn't set, "**vi**" will be used.
escape=c	Substitute the character *c* for the **~** escape character.
folder=directory	The directory for saving standard mail files. User-specified filenames beginning with a plus (**+**) are expanded by preceding the filename with this directory name to obtain the real filename. If *directory* doesn't start with a slash (**/**), **$HOME** is prepended to it. There is no default for *folder*. See also *outfolder*, below.
header	Enable the display of the header summary when entering **mailx**. Enabled by default.
hold	Preserve all messages that are read in the *mailbox* instead of putting them in the standard *mbox* save file. The default is *nohold*.
ignore	Ignore interrupts while entering messages. The default is *noignore*.
ignoreeof	Ignore end-of-file during message input. Input must be terminated by a period (**.**) on a line by itself or by the **~.** command. The default is *noignore*. See also *dot*.
keep	When the *mailbox* is empty, truncate it to zero length instead of removing it. Disabled by default.
keepsave	Keep messages that have been saved in other files in the *mailbox* instead of deleting them. The default is *nokeepsave*.

MBOX=*file* The pathname of a file to save message that have been read. The **xit** command overrides this function, as does saving the message explicitly in another file. The default is **mbox** in your home directory.

metoo If your login appears as a recipient, don't delete it from the recipient list. The default is *nometoo*.

LISTER=*command*

The utility (including options) to use when listing the contents of the *folder* directory. The default is the **ls -C** command.

outfolder Place the files used to record outgoing messages in the directory specified by the *folder* variable, unless an absolute pathname is specified. The default is *nooutfolder*. See the *folder* variable as well as the **copy**, **Followup**, **followup**, and **Save** commands.

page Used with the **pipe** command to insert a form-feed after each message sent through the pipe. The default is *nopage*.

PAGER=*command*

The utility (including options) to use as a filter for paginating output. The default is **more**.

prompt=*string* Set the command mode prompt to *string*. The default is an ampersand (&) followed by a space.

quiet Don't display the opening message and version when entering **mailx**. The default is *noquiet*.

record=*file* Record all outgoing mail in the named file. Disabled by default. See also *outfolder* above.

save Enable saving of messages in **dead.letter** on interrupt or delivery error. Enabled by default. For a description of this file, see *DEAD*.

screen=*number* Set the number of headers per page for the **headers** command. (Default: 20)

SHELL=*utility* The name of your preferred shell. The default is **sh**.

showto When displaying the header summary and if the message is from the user, the recipient's name, not the author's name, is displayed following **To**.

Sign=*string* The variable inserted into the text of a message when the **~A** autograph command is given. No default (see also **~i** in "Internal variables").

sign=*string* The variable inserted into the text of a message when the **~a** autograph command is given. No default (see also **~i** in "Internal variables").

toplines=*number* The number of lines of header to display with the **top** command. The default is 5.

VISUAL=*utility* The name of your preferred screen editor. Will default to the setting of the **VISUAL** environment variable. If the environment variable isn't set, "**vi**" will be used.

Environment variables:

HOME The pathname of your home directory.

MAILRC The pathname of a startup file. The default is **.mailrc** in your home directory.

The following environment variables influence the initial value of some **mailx** internal variables: **DEAD**, **EDITOR**, **MBOX**, **LISTER**, **PAGER**, **SHELL**, **VISUAL**.

© 2005, QNX Software Systems

Exit status:

0	Successful completion.
>0	An error occurred.

Syntax:

make [-einpqrst] [-f *makefile*]... **[-k | -S]**
[*macro*=*name*...] [*target_name*...]**

Options:

-e	Cause environment variables to override macro assignments within *makefile*s.
-f *makefile*	Use the description file *makefile*. If the pathname is the dash character (**-**), the standard input is used. If there are multiple instances of this option, they are processed in the order specified.
-i	Ignore error codes returned by commands. This is equivalent to the special **.IGNORE:** target.
-k	If a non-ignored error occurs while executing the commands to bring a target up-to-date, abandon work on the current target, but continue with targets that don't depend on the current target. This is the opposite of **-S**. If both **-k** and **-S** are specified, the last one specified is used.
-n	No-execution mode. Print commands, but don't execute them. However, lines preceded by a plus sign (**+**) prefix or containing the string **$(make)** *are* executed. Lines beginning with an **@** are printed.
-p	Write to standard output the complete set of macro definitions and target descriptions.
-q	Question mode. The **make** utility returns a zero or nonzero status code, depending on whether or not the target file is up-to-date. Targets aren't updated if this option is specified.
-r	Clear the suffix list and don't use the default inference rules.

-s	Be silent. Don't print command lines before executing them. This is equivalent to the special **.SILENT:** target.
-S	Undo the effect of the **-k** option. Stop processing when a nonzero exit status is returned by a command. If both **-k** and **-S** are specified, the last one specified is used. This option overrides the presence of the **k** flag in the **MAKEFLAGS** environment variable.
-t	Touch the target files, changing only the file date, rather than performing the rules to reconstruct them.
target_name	The target to bring up-to-date. If no target is specified, **make** uses the first target defined in the first *makefile* encountered.
macro=*name*	Macro definition. This definition remains fixed for the **make** invocation. It overrides any regular definitions for the specified macro within the *makefile*s and from the environment. This definition is inherited by subordinate **make** sessions but acts as an environment variable for these. That is, depending on the **-e** setting, it may be overridden by a *makefile* definition.

Description:

The **make** utility is used to maintain current versions of files by resolving time dependencies. If the file that **make** is examining (i.e. the target) is older than the file(s) on which it depends (i.e. the prerequisites), then **make** executes a list of shell commands associated with the target to create or update the target.

This document describes the POSIX-defined behavior of **make**. Specific **make** implementations will have extensions, for which you should consult the usage messages and original man page for the version of the utility you are using. To be guaranteed a portable makefile (one that will behave correctly under any POSIX-compliant **make** implementation), you should restrict the features used to those documented here, and you should put a **.POSIX** target in as the first non-comment line of the makefile. If this target is not put in, the behavior of **make** is implementation-defined and none of the information in this document necessarily applies (though typically it will apply regardless of the presence of **.POSIX**).

The **make** shipped with QNX is currently a port of GNU's **make** and therefore supports many extensions to the POSIX behavior. The full, original documentation for GNU **make** is available from the Free Software Foundation.

To control what **make** does, the utility references specifications (rules) consisting of a target, prerequisites, and a list of shell commands to be executed when a prerequisite is newer than the target. To simplify maintenance of programs, **make** has a collection of default macros and inference rules allowing it to identify the proper procedures required to update targets based on minimal information. (If the **-r** is given the builtin rules will not be used.)

Here's a simple example. Suppose you have a file named **myprog.c** and you wish to compile it to produce the executable program **myprog**. To do this, you would simply enter the command:

```
make myprog
```

© 2005, QNX Software Systems

This works even in the absence of a makefile. The alternative of using the **cc** utility would produce an executable named **a.out** by default. Using **make** for compiling single-source executables is a convenient shortcut.

The **make** utility attempts to perform the actions required to ensure that the specified target(s) are up-to-date. A target is considered out-of-date if it is older than any of its prerequisites. The **make** utility treats all prerequisites as targets themselves and recursively ensures that they are up-to-date. The utility examines the modified times of files to determine whether they are out-of-date.

After all the prerequisites of a target are ensured to be up-to-date, and if the target is out-of-date, the shell commands associated with the target entry are executed. If no shell commands are listed for the target, it is treated as up-to-date.

The section that follows is an alphabetical reference for **make** containing these topics: (numbers indicate the suggested order of reading for first-time users of **make**)

- Signals
- Default Rules and Macros (5)
- Inference Rules
- Internal Macros
- Macro Definition
- Macro Expansion
- Makefile Execution (4)
- Makefile Syntax (2)
- Recursive Make
- Special Macros

- Special Targets
- Suffix Rules
- Targets (3)
- Using Makefiles (1)

If you're a first-time **make** user, you should read the topics flagged with a number in the order given. By reading these sections, as well as the examples section, you should be able to construct simple, but useful makefiles. The other topics will help you create more powerful, flexible, and easier-to-maintain makefiles. If you're familiar with other **make** implementations, you should read all the topics in detail in order to use **make** to its best advantage and to avoid surprises.

Signals

If not already ignored, **make** traps SIGHUP, SIGTERM, SIGINT, and SIGQUIT and removes the current target unless it is a directory or a prerequisite of the special **.PRECIOUS** target. After cleaning up, **make** exits if the signal received was SIGQUIT. Otherwise, **make** resets the signal handler to the default and sends itself the signal that it received. The default action is taken for all other signals.

Default Rules and Macros

The POSIX default inference rules and macros for **make** consist of the the following:

If a user-supplied makefile redefines any of the default rules or macros, the result is unspecified.

The following briefly outlines the form of macro assignments and rule specifications. For more details, see "Makefile Syntax," "Macros," "Targets," and "Special Targets."

```
macro=value

target:prerequisite...
```

```
        shell command
        shell command
        .
        .
        .
```

The set of default rules will have the same effect as if the following rules were present:

Suffixes and Macros

```
.SUFFIXES: .o .c .y .l .a .sh

MAKE =     make
AR =       wlib

ARFLAGS =  -b -c
YACC =     yacc
YFLAGS =
LEX =      lex
LFLAGS =
LDFLAGS =
CC =       cc
CFLAGS =
```

Single Suffix Rules

```
.c:
        $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $<

.sh:
        cp $< $@
        chmod a+x $@
```

Double Suffix Rules

```
.c.o:
        $(CC) $(CFLAGS) -c $<

.y.o:
```

```
        $(YACC) $(YFLAGS) $<
        $(CC) $(CFLAGS) -c y.tab.c
        rm -f y.tab.c
        mv y.tab.o $@

.l.o:
        $(LEX) $(LFLAGS) $<
        $(CC) $(CFLAGS) -c lexy.yy.c
        rm -f lex.yy.c
        mv lex.yy.o $@

.y.c:
        $(YACC) $(YFLAGS) $<
        mv y.tab.c $@

.l.c:
        $(LEX) $(LFLAGS) $<
        mv lex.yy.c $@

.c.a:
        $(CC) -c $(CFLAGS) $<
        $(AR) $(ARFLAGS) $@ $*.o
        rm -f $*.o
```

Inference Rules

The **make** utility uses the suffixes of a target and its prerequisites to infer how a target can be made up-to-date. A list of suffix rules defines the commands to be executed (e.g. **cc -c** would typically be executed to turn a file ending in a **.c** suffix into a file ending in a **.o** suffix). By default, **make** contains a builtin set of inference rules. Additional rules can be specified in the explicit *makefile*, or any of the builtin rules can be redefined.

Inference rules are intimately tied to the **.SUFFIXES:** special target. Each entry in the **.SUFFIXES:** special target defines a suffix to a filename that may be used to build another file. The inference rules then define how to actually build one file from another. These files are related, in that they must share a common basename, but have different suffixes (e.g. **test.c** and **test.o**).

For each of the suffixes specified, **make** looks for a suffix rule. The suffix rule to update a target with a suffix of **.s1** from a prerequisite with a suffix of **.s2** is specified as a target **.s2.s1**.

For example, if the target suffix was **.o** and the source suffix was **.c**, the rule **.c.o:** specifies the command required to create a **.o** target file from a **.c** prerequisite.

When no explicit rule is found to update a target, the inference rules are checked. The suffix of the target (**.s1**) to be built is compared to the list of suffixes specified by the **.SUFFIXES** special target. If the **.s1** suffix exists, the suffix rules are searched in the order defined by the order of suffixes in **.SUFFIXES** for a **.s2.s1** rule. A check is made for the target name **$*.s2**. If this exists and if the file ending in **.s2** is out-of-date with respect to the file ending in **.s1**, then the commands for the **.s2.s1** rule are executed.

If the target to be built doesn't contain a suffix, and there's no rule for the target, the single-suffix rules are checked. The single-suffix rules define how a target is to be built if **make** finds a target with one of the single suffixes appended. A rule with one suffix (**.s2**) is the definition of how to build **target** from **target.s2** (e.g. the suffix rule **.c:** will be used to invoke the commands necessary to compile and link files ending in **.c** when a C source file is specified without its suffix and no rule is found, as in "**make test**").

Internal Macros

Several dynamically maintained macros are useful as abbreviations within rules. It's best not to define them explicitly.

$* The basename of the current target.

$< The name of the current dependency file.

$@ The name of the current target.

$? The names of prerequisites that are younger than the target.

$% Evaluated only when the current target is an archive library member of the form *libname*(*member***.o**). When this is the

case, **$@** evaluates to *libname* and **$%** evaluates to *member*`.o`.

The **$<** and **$*** macros are normally used for inference rules. They may not be reliable when used within explicit target command lines. The internal macros may be suffixed with **D** and **F** to specify the Directory and Filename components. For example:

```
${*D},${@F}
```

If there is no directory in the name, dot (`.`) is supplied.

Macro Definition

Macro definitions are in the form:

string1`=`*string2*

The value of *string2* is defined as all characters up to a comment character (#) or a **`newline`** character. The **`newline`** is ignored when preceded by a backslash (\). An escaped **`newline`** character in the file followed by any white space on the following line will be mapped to a single space.

Subsequent appearances of **`$(`***string1***`)`** or **`${`***string1***`}`** are replaced by *string2*. See "Macro Expansion" for more details. The parentheses or braces are optional if *string1* is a single character. The macro **$$** is replaced by the single character **$**.

Macro assignments come from the sources listed below. If a macro name already exists at the time it is being processed, the newer definition replaces the existing definition unless it is marked as unchangeable. The order is as follows:

1. macros specified on the command line are added and marked as unchangeable
2. macros defined in the **`make`** "builtin" rules (the **`default.mk`** file) are used

3 the contents of the environment are added to the macro table in the order defined in the environment

4 macros defined in the explicit *makefile*(s); multiple *makefile*s are processed in the order specified

If the **-e** option is present, the last two steps are reversed.

The *SHELL* macro is treated specially. It is provided by **make** and is set to the pathname of the Shell command language interpreter (**/bin/sh**). The *SHELL* environment variable is always ignored. If *SHELL* is defined in the explicit *makefile* or is specified on the command line, then it replaces the original value.

Macro Expansion

If you have defined a macro as follows:

*string1***=***string2*

then subsequent appearances of **$(***string1***)** or **${***string1***}** are replaced by *string2*. If *string1* is a single character, the parentheses or braces are optional.

Macros can appear anywhere in the makefile and are evaluated when used. Macros used in targets or prerequisites are evaluated when the makefile is being read. All other macros are evaluated when commands are executed. Macros used within macros are evaluated only when the new macro itself is evaluated. For example:

```
OBJSALL = $(OBJS1) $(OBJS2)
OBJS1 = a.o b.o c.o
OBJS2 = x.o y.o z.o
```

$*(OBJSALL)* will become **a.o b.o c.o x.o y.o z.o**

The optional form **$(***string1***[:***subst1***=[***subst2***]])** can be used to replace all non-overlapping occurrences of *subst1* with *subst2* when the macro substitution is performed. The occurrence of *subst1* must be a suffix at the end of a word in *string1*. It is as though the **sed** expression of the following form had been used:

s/\(.*\)*subst1***\([<blank><newline>]\)/\1***subst2***\2/g**

For example, using the macros defined above, you could create a list of source files (assuming they're all C source files) with the following:

```
SRCS = $(OBJSALL:o=c)
```

$*(SRCS)* will become **a.c b.c c.c x.c y.c z.c**

Makefile Execution

Once **make** has brought all the prerequisites up-to-date, it executes the list of specified shell commands. These commands are processed one line at a time. Command processing is done by writing the command to the standard output (except where noted otherwise) and then executing the command. Commands are executed by passing the command line to the shell in the same manner as the *system()* function.

The environment for the shell command being executed contains all of the variables in the environment of make. The macros from the command line to **make** and the *MAKEFLAGS* variable are added to the environment, overwriting any variable with the same name.

By default, when **make** receives a nonzero status from the execution of a shell command, it terminates with an error message to standard error.

All errors found while executing the command are ignored if any of the following conditions is true:

- the command prefix contains a hyphen (**-**)
- the **-i** option is present
- the special target **.IGNORE:** is in the *makefile*

The command is *not* written to standard output before it is executed if any of the following conditions are true:

- the command prefix contains an at-sign (**@**)

- the **-s** option is present
- the special target **.SILENT:** is in the makefile

If the **-n** option is present, **make** writes only the command line to standard output and does *not* execute the command. The only exceptions are if the command prefix contains a plus sign (**+**) or if the line contains the string **$***(MAKE)*.

The **-n** option is always added to **MAKEFLAGS**. This allows a recursive **make -n** *target* to be used to see all of the actions that would be performed on the target.

If the **-k** option is present, **make** terminates processing the commands for the current target if an error occurs executing one of the commands, but continues working on other targets that don't depend on the current target.

If the **-t** option is present, **make** writes the command line to standard output. But instead of executing the command, **make** updates the modified time of the target as though **touch** *target* had been executed. (See the **touch** utility.) This can be dangerous when sources are maintained by more than one person.

For more details on command lines and command prefixes, see "Targets."

Makefile Syntax

The *makefile*(s) may contain a mixture of comment lines, macro definitions, include lines, and target lines including special targets and suffix rules. Lines may be continued across input lines by escaping the **newline** with a backslash (\).

Anything after a pound sign (**#**) is considered a comment and is stripped from the line, including spaces immediately before the #. If the # is inside a quoted string, it isn't treated as a comment. Completely blank lines are ignored.

An include line is used to include the text of another makefile. It consists of the word **include** with no leading blanks, followed by

spaces and by the name of the file to be included at this line. Macros in the name of the file being included are expanded before the file is included. Files that are included may be nested. An example of an include line is:

```
include my_defaults.mk
```

For more details on the other different line types, see the following topics: “Targets,” “Macros,” “Special Targets,” and “Suffix Rules.”

Recursive Make

The **make** utility has the ability to invoke itself. This can be extremely useful for special circumstances, such as making a library in another directory before linking a program that requires it.

When **make** is processing commands, if the command prefix contains a plus (**+**) the command will always be executed, even if the **-n** option is present. See “Targets” for details.

Here’s a commonly seen makefile construct that demonstrates the use of a recursive make:

```
program:    libdir/example.lib
    ...

libdir/example.lib:
    +cd $(@D);$(MAKE) $(@F)
```

Special Macros

MAKEFLAGS This macro has the set of options provided to **make** as its value. If this is set as an environment variable, the set of options is processed before any command-line options. This macro may be explicitly passed to nested **make**s, but it is also available to these invocations as an environment variable. The **-f** and **-d** flags aren’t recorded in this macro.

$	This macro translates to a dollar sign. Thus you can use **$$** in the *makefile* to represent a single $.

Special Targets

.DEFAULT:	The default is used to process a target when there is no other entry for it, and no implicit rule for building it. The **make** utility ignores all prerequisites for the default target.
.IGNORE:	Nonzero error codes returned from commands are ignored. Specifying this in a *makefile* is the same as specifying **-i** on the command line.
.POSIX	This target must be specified before the first non-comment line in the makefile to instruct the **make** implementation to interpret the file according to the POSIX behavior documented here.
.PRECIOUS:	Prerequisites of this special target aren't removed if **make** receives a signal. Subsequent occurrences of **.PRECIOUS:** add to the list of precious files. There are no commands associated with the **.PRECIOUS:** special target.
.SILENT:	Commands aren't echoed before they're executed. Specifying this in a *makefile* is the same as specifying **-s** on the command line.
.SUFFIXES:	The suffixes list for selecting implicit rules. Specifying this target with prerequisites adds these to the end of the suffixes list. Specifying it with no prerequisites clears the list. To change the order of the suffixes list, you must specify a suffixes target with no prerequisites (clear the list) then specify a suffixes target which specifies the suffixes in the desired order.

Suffix Rules

Targets that begin with a period and don't contain any slashes are treated as suffix rules. If there is only one period in the target, it is a single suffix rule. Targets with two periods are double suffix rules.

Suffix rules don't have prerequisites, but do have commands associated with them. See "Inference Rules." For examples of the default rules, see "Default Rules and Macros."

Suffix rules can be redefined or canceled. A target that matches an existing suffix target overwrites the old suffix rule. If no commands are given, the rule is never used.

Targets

A target entry in the *makefile* has the following format:

target`... : [`*prerequisite*`...] [;` *command*`]`
`tab`*shell_command*
`tab`*shell_command*
`.`
`.`
`.`

Any line that doesn't have leading tab (other than macro definitions) is a "target" line. Target lines consist of one or more filenames (or macros that expand into same) called targets, followed by a colon (`:`). The colon is followed by a list of prerequisite files. The prerequisite list may be terminated with a semicolon (`;`), which may be followed by a shell command.

A line in a *makefile* that starts with a **`tab`** character is a shell line or rule. This line is associated with the current target entry. A sequence of commands may be associated with a single target entry. Shell lines may have any combination of the following characters to the left of the command as a *command prefix*:

@ won't echo the command line, except if **`-n`** is used

\- **`make`** will ignore the exit code of the command; without this, **`make`** terminates when a nonzero exit code is returned

+ **make** executes the command even if **-n** was specified

Only one target rule for any given target may contain commands. However, if the same target is named in multiple target rules, the prerequisites named will be added together to form a complete prerequisite list.

The prerequisites are those from which a target is constructed. They in turn may be targets of other prerequisites. In general, for a particular target file, each of its prerequisite files is "made" to ensure that each is up-to-date with respect to its prerequisites.

The modification time of the target is compared to the modification time of each prerequisite file. If the target is older than the prerequisite, then one or more of the prerequisites have changed, and the target must be constructed. This checking is done recursively, so that all prerequisites of prerequisites of prerequisites *ad infinitum* are up-to-date.

To reconstruct a target, **make** expands macros, strips off initial white space, and passes each rule to a shell for execution.

For target lines, macros are expanded as the *makefile* is read. Command lines have macro expansion delayed until the line is to be executed.

Using makefiles

The **make** utility comes with a set of builtin macro definitions and rules. If the **-r** option is specified, the "builtin" rules aren't used and the suffix list is cleared.

You can specify additional rules in a user-specified *makefile*. Targets, rules, and macros of the same name are overwritten by the user-specified file. By default, the **./makefile** file is used. If **./makefile** isn't found, the file **./Makefile** is then tried.

The **-f** option directs **make** to ignore **./makefile** and **./Makefile** and use the specified argument as a *makefile* instead. If the **-** argument is specified, standard input is used. More than one additional *makefile* can be used by specifying multiple **-f** options.

Using **-f** multiple times performs a concatenation of all the *makefiles* in a left-to-right order.

See "Makefile Syntax" for more details.

Examples:

```
make test.o
```

Typing this in the absence of a makefile (or with a makefile that doesn't mention the **test.o** file) will use the builtin suffixes list and inference rules to determine the name of the source file and, if necessary, to run the proper command to create or rebuild the **test.o** file.

Suppose you have the source files **test1.c**, **test2.c**, and **test3.c**. The first two files include the headers **<hdr1.h>** and **<hdr2.h>**, and these files are all linked together to produce the program **myprog**. Here's what a simple makefile describing this could look like:

```
#  Samplemakefile1

myprog:  test1.o test2.o test3.o

test1.o:  hdr1.h hdr2.h

test2.o:  hdr1.h hdr2.h

test3.o:
```

You could compile and link **myprog** by entering:

```
make myprog
```

Or since **myprog** is the first target, you could simply enter:

```
make
```

If you wanted to see what commands need to be executed without actually executing them, you could type:

```
make -n
```

Using macros, the **myprog** makefile could be simplified to:

```
#  Samplemakefile2

OBJS=test1.o test2.o test3.o
HDRS=hdr1.h hdr2.h

myprog:  $(OBJS)

test1.o:  $(HDRS)

test2.o:  $(HDRS)
```

This makefile is functionally identical to the previous example. It's not significantly better, just slightly more generalized.

We can further simplify the makefile by using the builtin inference rules and macros, as follows:

```
#  Samplemakefile3

OBJS=test1.o test2.o test3.o
HDRS=hdr1.h hdr2.h

myprog:  $(OBJS)

test1.o test2.o: $(HDRS)
```

As you can see, this makefile is short and to-the-point. Again, this is functionally equivalent to the previous examples.

Using this makefile, you can customize the compilation from the **make** command line by setting the appropriate macros:

```
make CFLAGS=-Oaxt
```

which increases the optimization level of the compile.

You can also direct the linker to produce a linkmap:

```
make LDFLAGS=-M
```

Of course, any of these macro assignments could be "hard-coded" in the makefile, but it's often convenient to override the defaults from the command line for special needs.

Environment variables:

MAKEFLAGS

This environment variable, if it exists, is read by the **make** utility, and is treated as a set of option characters to be used as the default options. Command-line options to **make** have precedence over these options.

After **make** has started and the **MAKEFLAGS** variable has been read, all of the options except **-d**, **-f**, and **-p** are added to the *MAKEFLAGS* macro. The *MAKEFLAGS* macro is passed into the environment as an environment variable for all child processes.

SHELL The value of the **SHELL** environment variable is always ignored. All other environment variables are used as macros as defined in the section "Macro Definition," above.

Exit status:

When the **-q** has been specified, the exit status has the following significance:

0 Successful.

1 The target was not up-to-date.

>1 An error occurred.

When the **-q** has not been specified, exit status is

0 Successful.

>0 An error occurred.

© 2005, QNX Software Systems

See also:

gmake

Andrew Oram and Steve Talbott, *Managing Projects with make*, O'Reilly and Associates, 1991.

Syntax:

melt [-cfvV] [*filename***...]**

Options:

See **freeze**.

Description:

The **melt** utility is equivalent to **freeze -d**. See the **freeze** utility (Compress and uncompress files) for details.

The **freeze**/**melt**/**fcat** compression utilities will eventually become deprecated in favor of the GNU **zip** suite, **gzip**/**gunzip**/**zcat**. The **freeze** suite of utilities will continue to be provided for quite some time before being eliminated completely.

Author:

Leonid A. Broukhis

See also:

freeze, **gzip/gunzip/zcat**

mkcis

© 2005, QNX Software Systems

Create a Card Information Structure (QNX)

Syntax:

mkcis [-2] [-f *outfile***] [-o** *outfile***] [-q] [-w** *level***]** *file*

Options:

-2	Output level 2 tuples only.

Don't use the **-2** option if you're using **Efsys.file**. **Efsys.file** can't read the level 2 structures.

-f *outfile*	Generate an image of the Flash device in *outfile*.
-o *outfile*	Output the CIS to this file (default is standard output).
-q	Be quiet.
-w *level*	Set the warning level.

Description:

The **mkcis** utility reads a file containing a text description of the memory regions and partitions that make up an EFS socket. The utility attempts to find the optimal placement of partitions and creates a CIS containing this information. An Initial Program Loader (IPL) may precede the CIS on embedded filesystems used for booting.

The input file is made up of a sequence of sections. A section can be a description either of the geometry of a memory device or of a region. A region section may contain any number of partition sections. You determine the placement of each partition by assigning various attributes to it. These attributes tell **mkcis** the types of accesses that will be made and the alignment restrictions that the partition must conform to.

The input file consists of lines in this format:

[spaces/tabs] *keyword* **[***value***]**...

Prefix every comment with the **#** character, which causes all input to the end of line to be ignored.

You can specify numbers in decimal (the default) or in hex by prefixing them with **0x**. A number can have **M** or **K** suffixes to signify multiplication by 1024×1024 or by 1024. The case of these suffixes doesn't matter.

Card Information Structure

Here's the file structure:

```
jedec
    id number
    type [dram|sram|flash|eprom|eeprom]
    size number
    write_disrupt number
    erase_disrupt number
    erase number
    write number

region
    jedec number
    size number
    offset number
    interleave number
    boot_file "filename"
    partition
       type [efs|image]
          file "filename"
          size number
          attribute [erase|write|execute|write_align|
                     erase_align|largest]*
```

jedec section

The **jedec** section defines the type and geometry of a memory device used in the EFS. Standard device definitions can be found in the file **/etc/jedecs** and are read by **mkcis** at startup. You can specify the following attributes:

id Memory devices are identified by a 16-bit **jedec** number. ROM, DRAM, and SRAM devices don't need **jedec** information.

type The type of memory device, which can be one of **dram**, **sram**, **flash**, **eeprom** or **eprom**.

size The size of the device in bytes.

write_disrupt

When a memory device is performing a write operation, access to other memory locations may be disrupted until the write has completed. For SRAM devices this value is usually **1**, meaning that only the location being written to is inaccessible. For many Flash devices the entire chip is unavailable for reads until the write has completed. For example, the Intel 28F008 chip has a **write_disrupt** value of **1M**.

erase_disrupt

Same as above but for erase operations. For example, the Intel 28F008 chip has an **erase_disrupt** value of **1M**.

erase The minimum number of bytes that can be erased in a single operation (the erase block size).

write The minimum number of bytes that can be written in a single operation.

region section

A region section must be defined for each region in the EFS. Each section contains the following attributes:

jedec The **jedec** number of the memory devices in the region.

size The size of the region in bytes.

offset The offset in bytes of the start address of the region from the start of the EFS (default is 0).

interleave The number of devices that the region is interleaved across.

boot_file The name of a binary file containing an IPL. The IPL code is placed before the CIS on the EFS.

Each partition in a region has the following attributes:

type The partition type, which can be **efs** or **image**.

file The name of a binary file containing an image to be placed in the partition. Usually generated by **mkffs** or **romqnx**.

size The size of the partition in bytes.

attribute The functional attributes of the partition, including:

- **write** The partition will be written to.
- **erase** The partition will have erase operations.
- **write_align** The partition must be aligned on a write block boundary.
- **erase_align** The partition must be aligned on an erase block boundary.
- **xip_align** The partition should be aligned on a 4K boundary.
- **largest** The partition should take up the largest free area in the region.

When specifying an embedded filesystem with a **file** directive, **attribute largest** and **size** are ignored and shouldn't be specified.

Examples:

Create a card information structure (CIS) from input file **exm2a.info** and write the CIS to raw device **/dev/skt1**:

```
mkcis -o /dev/skt1 -f exm2a.img exm2a.info
```

Here's a sample CIS information file for an Octagon 5025A system with a single EFS consisting of Intel 28F020 Flash memory devices. The EFS is configured to contain only a bootable image partition.

```
#
# Configure CIS for Octagon 5025A
#
jedec
    id 0x89bd
    type flash
    write_disrupt 256k
    erase_disrupt 256k
    erase 256k
    write 1
    size 256k

region
    size 256K
    interleave 1
    jedec 0x89bd
    boot_file "/boot/sys/boot.5025a"

partition
    type image
    attribute largest

partition
    type ffs
    file "ffs.img"
```

This input file creates a CIS suitable for an Intel EXPLR2 board:

```
#
# Sample configuration for EXPLR2
#
region
    size 2M
    jedec 0x89A0
    boot_file "/boot/rom.explr2"
```

```
partition
    type image
    file "nto.image"
```

See also:

mkffs, **mkipl**, **romqnx**

mkdir

© 2005, QNX Software Systems

Make directories (POSIX)

Syntax:

mkdir [-m *mode***] [-p] [-s** *size***]** *dir***...**

Options:

-m *mode* When creating the directory, set the file permission bits of the new directory to the specified *mode* value.

The *mode* argument is a *symbolic_mode* string, as defined for the **chmod** utility. In the *symbolic_mode* strings, the *op* characters **+** and **-** are interpreted relative to the default file mode for that file type:

- **+** add permissions to the default mode
- **-** delete permissions from the default mode
- **=** assign permissions

-p Create any missing intermediate pathname components.

-s *size* Create the directory with the given initial size (where the size is the number of entries a directory will hold before it has to be grown). This will reduce the number of filesystem extents a file will occupy if it is filled with a large number of files. Note that large directories should generally be avoided since they will increase the time it takes to open a file in that directory (and in particular, the time it takes to create a new file) since the filesystem will have a larger directory to scan when looking up the file by name.

dir A pathname at which a directory is to be created.

If you specify both the **-p** and **-m** options, any intermediate directories you have created have mode **u+wx**.

Description:

The **mkdir** utility creates the directories specified by the *dir* operands, in the order the *dir* operands are specified.

To create a directory you must have write permission on the parent directory, or be **root**.

The default file mode for directories is **a=rwx** (777), with selected permissions removed in accordance with the file mode creation mask (see the **umask** utility).

For intermediate pathname components created by **mkdir**, the mode is the default modified by **u+wx** so that the subdirectories can always be created regardless of the file mode creation mask. If you want to assign different ultimate permissions for the intermediate directories, you can do so with the **chmod** utility.

When using **-p** with **-m**, each intermediate directory that doesn't exist is created with **u+wx** modes, regardless of the file mode creation mask. The specified *mode* applies only to the last directory specified. For example:

```
mkdir -p -m 777 dir/dir1/dir2
```

would give **dir** and **dir1** the default permissions for intermediate directories (namely, **u+wx**). The directory **dir2** is given **a+rwx** permission.

The default file creation mask influences the behavior of **mkdir**.

Examples:

Create a directory named **/home/debbie**:

```
mkdir /home/debbie
```

© 2005, QNX Software Systems

Signals:

If the **mkdir** utility is terminated by a signal, some of the specified directories may have already been created.

Exit status:

0 All the specified directories were created successfully, or the **-p** option was specified and all the specified directories now exist.

>0 An error occurred.

See also:

chmod, **rmdir**, **umask**

Syntax:

mkffs [options] -o *fname root_directory*

Options:

-o *fname* Write the output of **mkffs** to *fname*.

-b *bsize* Specify the block size of the component blocks in the Flash device. This value is specified in bytes or kilobytes. The default value is 64K.

-c Compress all files.

-m[anu] Translate case for filenames:

- **a** always convert to lowercase
- **n** never convert to lowercase
- **u** convert uppercase-only filenames to lowercase

The default for QNX is **-mn**; for Windows NT, it's **-mu**.

-p Select POSIX mode. In this mode, user IDs and group IDs are stored for the filename directories in the Flash device, in accordance with POSIX.

-r *spare* Keep *spare* spare blocks for use during reclamation (default is 1, the maximum is 8). If you specify 0 spare blocks, the embedded filesystem is read-only.

There's no real advantage to having more than one spare block.

-s *fsize* If the resulting embedded filesystem image file is less than *fsize*, fill the rest of the image file with **0xFF** characters and appropriate format to fill it to *fsize* bytes. This can be specified in bytes or kilobytes.

-q Be quiet — warnings aren't displayed.

-v Be verbose.

Description:

The **mkffs** utility creates a Flash (that is, embedded) filesystem image in the specified *fname*, based on the files contained in the *root_directory* parameter. The optional *bsize* parameter is used to specify the block size of each erasable subdevice within the embedded filesystem.

Note that the directory structure created is of the entire specified directory *and any subdirectory entries*. The specified directory becomes the root of the created embedded filesystem.

When building an image with **mkffs**, symbolic links found in and below the root directory are ignored. These links and/or the files they refer to aren't included in the image. To include these files in the image, use a hard link or the file itself.

The **-s** *fsize* flag is used to fill the embedded filesystem image with **0xFF**s and appropriate formatting information. If the embedded filesystem exceeds *fsize*, **mkffs** exits with an error.

Unless at least one spare block is specified (using the **-r** option), the filesystem won't be able to reclaim space in other blocks.

Examples:

To create an embedded filesystem image called **flash_1**, from the files contained in the directory **/home/jwall/flash**:

```
mkffs -o flash_1 /home/jwall/flash
```

To create an embedded filesystem image, as above, specifying that the erase block size on the device is 64K:

```
mkffs -b 64k -o flash_1 /home/jwall/flash
```

See also:

mkcis, **mkipl**, **romqnx**

mkfifo

© 2005, QNX Software Systems

Make FIFO special files (POSIX)

Syntax:

mkfifo [-p] [-m *mode***]** *file***...**

Options:

-m *mode* When creating the FIFO special file, set the file permission bits of the new file to the specified *mode* value.

The *mode* option argument is a *symbolic_mode* string, as defined for the **chmod** utility. In the *symbolic_mode* strings, the *op* characters **+** and **-** are interpreted relative to the default file mode for that file type, as follows:

- **+** add permissions to the default mode
- **-** delete permissions from the default mode
- **=** assign permissions

-p Create directories in the FIFOs pathname, if required.

file The pathname at which a FIFO special file is to be created.

Description:

The **mkfifo** utility creates the FIFO special files specified by the *file* operands in the order they're specified.

To create a FIFO in a directory, you must have write permission for that directory or be logged in as **root**.

The default file mode for FIFO files is **a=rw** (666) with selected permissions removed in accordance with the file mode creation mask (see **umask**). For intermediate pathname components created by **mkfifo**, the mode is the default modified by **u+wx** so that any subdirectories and the FIFO file can always be created regardless of the file mode creation mask. If you wish to assign different ultimate permissions for the intermediate directories, you can change these permissions afterward with the **chmod** utility.

Signals:

If the **mkfifo** utility is terminated by a signal, some of the specified FIFO special files or intermediate directories might have already been created, and may not be automatically removed.

Exit status:

0 All the specified FIFO special files were created successfully.

>0 An error occurred.

See also:

chmod, **ln**, **mkdir**, **touch**

mkipl

© 2005, QNX Software Systems

Convert a binary image into ROM format (QNX)

Syntax:

mkipl [*options***...** *imagefilename***]**

Options:

-a *align*[**K|M|G**]

Force the next file to be aligned on an *align*-byte boundary (the default is 1 byte).

-f *format* Output format:

binary or **b**	raw binary file
srec or **s**	Motorola S records
intel or **i**	Intel hex records

The default is **srec**. When using **binary**, the file's offset is printed to *stderr*.

-l *reclen* Length of data bytes per line (the default is **32**)

-o *offset* Offset (the default is **0**)

-r Suppress the reset vector record.

-s *romsize*[**K|M|G**]

Size of ROM (the default is **4G**). The case of the suffixes doesn't matter.

Description:

The **mkipl** utility converts a binary image into either Motorola S records or Intel hex records, which are suitable input for most Flash/EPROM programmers. Most ICEs also expect these formats. The format defaults to Motorola S records but may be changed using the **-f** format option. Note that the Intel format is limited to 1M offsets.

By default, **mkipl** assumes the image contains code that is the initial program loader (IPL), which is connected to the power-on reset vector of the processor. The **mkipl** utility locates the image such that it's placed in the address space where it will end at the reset vector. A record is then output for the reset vector, which will do a 16-bit relative **jmp** to the start of the image.

The reset vector on an Intel 386/486/... processor is located at linear address **0xFFFFFFF0** (16 bytes below 4 Gigabytes). The offsets used for the records by default address this area. It's as though you had a 4-Gigabyte ROM and needed to load code into the top of it. This is what most ICEs expect. If you're sending your output to a small Flash/EPROM programmer, it may wish the offsets to be relocated to the top of the the Flash/EPROM alone. This can be accomplished using the **-s** *size* option.

If the **-r** option is set, it indicates that the image should not be considered a power-on IPL. The record offsets for the image start at zero; a **jmp** won't be programmed at the top of the device. The record offset can be changed using the **-o** option. Note that the **-o** option is used only if **-r** is specified.

For compatibility with most devices that accept these records, each record is limited to a maximum of 32 bytes. This can be increased to a maximum of 255 bytes using the **-l** option. Larger records have less overhead and result in slightly faster downloads.

Examples:

The following converts the binary image **explr2** into Motorola S records. The offset used in the records starts the image such that it ends at the reset vector **0xFFFFFFF0**. A 16-bit relative **jmp** is programmed at **0xFFFFFFF0** to **jmp** to the start of the image.

```
mkipl explr2
```

The following converts the binary image **ipl** into Intel hex records. The offset used in the records starts the image such that it ends at the reset vector in the 256K ROM (**0x3FFF0**). A 16-bit relative **jmp** is programmed at **0x3FFF0** to **jmp** to the start of the image.

```
mkipl -f i -s 256k ipl
```

The following converts the binary image **notipl** into Motorola S records. The offset used in the records starts at **0**; a reset vector **jmp** isn't output.

```
mkipl -r notipl
```

See also:

mkcis, **mkffs**, **romqnx**

Syntax:

modem [*options***]**

Options:

-a *string*	Use *string* to answer the phone (default is **ATA**).
-b *baud*	Use this baud rate when programming the modem or receiving result codes (default is 2400 baud).
-B *baud*	Assume this baud rate if the modem answers "connect" without a specified baud rate (default is 2400 baud).
-c *command*	Start this command (default is **login -p**).
-C	Don't wait for CTS.
-d [**P**]*number*	Dial this number and attempt to connect to a remote modem. If the number is preceded by a **P**, then pulse dialing as opposed to tone dialing will be used. Upon a successful connection, **modem** will attempt to login to the remote system if the **-U** *user***:***pass***:***cmd* option is specified. It will then *exec()* into either the command specified by the **-c** option or **/bin/qtalk** if **-c** is not specified.
-D *file*	Log debug information. If *file* is a dash character (**-**), then output is sent to *stderr*. (Default: no logging of debug info)
-f *command*	Command to start for a fax call. (default: **/bin/fax re**)
-F *fax_id*	Local fax machine identification. (default: "Unknown")
-g *file*	On connect, print the greeting message stored in *file*.

© 2005, QNX Software Systems

-i *string* Use this string to initialize the modem (default is **ATZ**).

-I *file* Identification file for caller-id/callback processing.

-l ("el") Do not lock the baud rate.

-L Use locked baud rate. (default)

-p *parity* Set the parity, number of data bits, and number of stop bits specified in *parity*. This consists of a three-character string:

parity	data	stop
{e|o|m|s|n}	{7|8}	{1|2}

The default is **n81**.

-P *millisec* Wait this number of milliseconds before responding to a **RING** with the answer string (usually **ATA**). The default is 0.

-r *rings* Wait for this number of rings before answering (default is 1 unless **-I** is specified, in which case the default is 2). To suppress answering, specify **-r 0**.

-R *minutes* Resynchronize the modem by sending the initialization string (specified via option **-i**) every specified number of *minutes* minutes (default is every 15 minutes). Specify **-R 0** to disable resynchronization.

-s *seconds* Delay this number of seconds before starting command (default is 3 seconds).

-t *tries* Number of times to try and dialout or callback. (default: 2)

-T *pppd_cmd* Command to start on a **ppp** call (default: none)

-U *user*:*pass*:*cmd*

Userid and password to login as. Only valid with **-d** option.

Description:

The **modem** utility monitors a serial line connected to a modem. It is usually started by the **tinit** utility to automatically provide dial-in users with a login.

For testing, you can start **modem** on a serial line using I/O redirection in place of the **tinit** utility. This will operate **modem** on a one-shot basis. That is, when a call is answered by the **modem** utility, it will start a **login** (this can be changed with the **-c** option) and terminate. When the **login** user logs out, **modem** will not be restarted. For example:

```
on -t /dev/ser1 modem -b 57600 -L -D debug_output
```

The **-D** *filename* option will write all the communication that occurs between the **modem** utility and the modem to the named file.

To configure a modem, the **modem** utility can be told to send commands to the modem. It will then wait for the modem to issue messages concerning an incoming call.

After **modem** has answered the phone and determined the baud rate, it waits for the delay specified via the **-s** option. This delay avoids the burst of noise characters that often occur when two older modems first connect. (On some ancient modems, this has been observed to be up to 10 seconds.) After the delay, a greeting is displayed, provided the **-g** option was used to specify a greeting file. The **modem** program will then replace itself via the *exec()* function call with either the **login** utility or the command specified with the **-c** option. The user may now log in as usual. If the user logs out, **tinit** restarts another **modem** program on the line and the whole process is repeated.

You can periodically reprogram a modem via the **-R** option. This is useful if the modem is powered off, then on again. This practice might also improve the reliability of some modems that "forget" their settings over time (this isn't supposed to happen in a perfect world...).

The **-p** option sets the line parameters for the parity, number of data bits, and the number of stop bits. It consists of a three-character string. The following table summarizes the five most common configurations. Note that most modern systems expect **n81**, the default.

Configuration	parity	data bits	stop bits
n81 (default)	none	8	1
e71	even	7	1
o71	odd	7	1
m71	mark	7	1
s71	space	7	1

To talk to another QNX system, **n81** should be selected.

The **-L** option should be used with high-speed modems that talk to the computer at a fixed baud rate and use hardware flow control to regulate the data. This option will enable QNX hardware flow control. Make sure you program your modem to support hardware flow control as well. Most modems support hardware flow control by default.

So that **modem** can detect the modem status (on or off line), you have your modem configured to provide the CD (carrier detect) RS-232 signal to the computer. A few modems do not support this capability, but most modern modems are configured for this by default. For older Hayes modems, the command to enable this is **AT&C1**, but check the manual for your modem since this command does vary. When a loss of carrier occurs, a SIGHUP is sent to the session leader, typically the **login** shell. This will cause it to terminate, which will send a

SIGHUP to non-background children. The net effect is to remove all the users' processes, thus logging the user off.

So that **modem** and other programs can force the modem to hang up if necessary, the DTR (**D**ata **T**erminal **R**eady) RS-232 signal should also be connected from the computer to the modem. Your modem should be programmed to force a hangup and reset itself when the computer "drops" DTR. Most modern modems do this by default. For older Hayes modems, the command to do this is **AT&D2**, but check the manual for your modem since this command varies.

As part of the baud rate determination, **modem** posts an environment variable called **BAUD**, which is set equal to the baud rate for the modem connection. The **BAUD** variable is useful since many high-speed modems operate with the serial link between the modem and the computer at a fixed high speed (usually 19200 or greater), and then use hardware flow control to regulate the flow of data for slower connections. Since the true baud rate cannot be determined by the **stty** setting, **modem** posts this information as an environment variable so that the tasks started by the dial-up user can determine the true transfer rate.

To change the modem initialization string used by **modem**, you use the **-i** option. The default string is "**ATZ**".

The modem initialization string supports a number of special characters that are processed by **modem** rather than being passed to the modem. These characters, and their actions, are:

Character	Action
|	carriage return
~	(tilde) 1-second delay
'	(apostrophe) 100-millisecond delay
^	(caret) drop DTR for 1 second (forces a modem to hang up and reset)

continued...

Character	Action
!	send a 500-millisecond break

As the modem initialization string is processed, any characters not in this special set are passed through to the modem, unmodified. Using these special characters, modem initialization strings that perform many steps can be easily constructed. For example:

```
^~~~ATZ
```

would force a hangup of the modem with DTR (^), delay 3 seconds (~~~) and then command the modem to reset to its saved state (**ATZ**, supported by most modems).

To accept incoming calls, your modem need not be programmed for auto-answer. The modem can be set to "not auto answer" (**ATS0=0**). The modem will emit "**RING**" statements when an incoming call is detected and **modem** will detect the "**RING**" and provide the answer string (default is "**ATA**") to force the modem to answer. This has the added advantage that your system won't answer the phone unless it is powered up *and* the **modem** program is ready for the incoming call. We recommend operating your modem in this mode.

We also recommend you default your serial ports to 8 bits, no-parity, and 1-stop bit for communication with other QNX systems.

Caller-ID and callback

If the **-I** option is specified modem will enforce caller-id authentication. Before being presented with a login prompt, **modem** will prompt you for a callback id. You will enter a name which **modem** will lookup in the callback file specified by the **-I** option. If it finds a match it will hangup, and callback the number associated with the name. For the person dialing in from a QNX system, they will typically be in **qtalk** and should type **ATA** to answer the phone when it rings. They will then connect to the remote **modem** utility which will immediately *exec()* into **login** as it would have if callback was not specified.

If you have a modem which supports caller-id and you have subscribed to this service with your telecom provider then **modem** will detect the special modem message "**nmbr = number**" and lookup the number on the callback file. If it finds a match, **modem** will not challenge you for a callback name but will *exec()* directly into **login**. Callback files consist of a number of one-line callback entries, each of which is in the format

symbolic_name: *dial_command phone_number*

A typical callback file follows:

```
tom: atdt 8385866
dick: atdt 7398837
harry: atdt1 7059420088
```

The symbolic name is terminated with a colon. The number to dial consists of two parts. The first part is the command necessary to cause your modem to dial. The second part is the part which is matched against the caller id sent by the telecom company. Note that caller-id strings do not have the leading 1 required to make a long distance call. That is why in the above example the 1 is included as part of the modem dial command. The same would be true if a 9 was required to dial out.

If a call fails caller-id authentication, the number is logged to a file with the same path as specified by the **-I** option with a "**.bad**" concatenated to the end. For example if **modem** was run with a **-I /etc/config/callback** option, then

/etc/config/callback

would be the callback file

/etc/config/callback.bad

would be the failed callback attempt logfile

This logfile can be used to determine the correct number to use when caller-id is enabled on a line, as well as for determining the identity of unauthorized callers.

Automatic login on callback

When dialing out, the **-U** option can be used to automatically log in. After a successful connect, if **-U** is specified **modem** will wait for a "login:", "password:" and "$" or "#" issuing the appropriate *user*, *pass* and *cmd* specified on the command line.

Fax calls

Modem will detect a fax call and by default invoke the program **/bin/fax** to process it. This can be changed by using the **-f** option. Note that although modem can detect a fax call it has no code to receive a fax itself. The program **/bin/fax** is a suggested name and is not shipped with the operating system.

PPP calls

Modem will detect a TCP/IP PPP call if this option is set and start the specified ppp daemon. This will introduce a 2 second delay in the answering of all calls as modem waits for a potential ppp connect. If you use this option you should make sure PPP enforces one of the authorization protocols.

Log File

The **modem** utility will log connect messages to the file **/etc/acclog**, provided the file exists. For details, see the QNX *Installation & Configuration*.

Examples:

Note that normally you would start each of these commands via the **tinit** utility, as opposed to directly from the command line. See "Connecting Character Devices" in QNX *Installation & Configuration*.

Handle calls on a 57600 baud modem that supports a locked baud rate and hardware flow control between itself and the computer:

```
modem -g /etc/modem_msg -b 57600 -L
```

Handle calls on a 57600 baud modem (as above), but print diagnostic information to console 9 for troubleshooting:

```
modem -b 57600 -L -D /dev/con9
```

Handle calls on an ancient 1890's brass-and-magnet 2400 baud formerly-state-of-the-art-but-now-a-museum-piece modem ensconced in the manufacturer's original equipment giltwood curio cabinet:

```
modem -l -b 2400
```

Files:

/etc/acclog (accounting records are appended to this file)

callback_file A file of authorized users, a callback command and their caller-id or callback numbers can be specified using the **-I** option. See the section "Caller ID and callback".

*callback_file***.bad**

When **-I** is used, a logfile of failed callback or caller-id authorization attempts will be maintained. The name of this file is the name of the callback file with a **.bad** suffix.

See also:

login, **qcp**, **qtalk**, **tinit**

monitor

© 2005, QNX Software Systems

Generate kernel event log (QNX)

Syntax:

monitor [-b *numbufs***] [-f** *file***]**
[-i *intr***] [-n] [-p** *priority***]**

Options:

-b *numbufs*	Number of buffers (4096 bytes/buffer, default: 8 buffers).
-f *file*	Save data to this file (default: **default.mon**).
-i *intr*	Log events for this interrupt (multiple **-i** options may be specified).
-n	Log all interrupts (default is none)
-p *priority*	Run at this priority.

Description:

The **monitor** utility writes a log of kernel events to a logfile which may then be interpreted for display by the **msgprint** utility.

Examples:

Generate a kernel event log in the default file **default.mon**:

```
monitor
```

Stop the event logging by terminating **monitor** (e.g. you could type Ctrl – C).

See also:

msgprint

Syntax:

more [-ceisu] [-n *number***] [-p** *pattern***]**
[-/ *pattern***] [-t** *tag***] [-x** *tabstop***] [***file***...]**

Options:

-/ *pattern*	Same as **-p** *pattern*.
-c	For each full screen of text that is displayed, clear the screen from the first line and display the next full screen of text.
-e	Stop after displaying the last line in the file. If the next command that displays text causes **more** to reach end-of-file again, **more** exits. If the file is shorter than a single screen, **more** exits at end-of-file regardless.
-i	Ignore case during searches. Uppercase and lowercase letters are considered identical.
-n *number*	Specify the number of lines that constitute a full screen of text. The *number* argument is a positive decimal integer. The **-n** option overrides any values obtained from the environment, and suppresses line numbering.
-p *pattern*	Search for a line that matches *pattern*. The current position is set to the first matched line. If no match is found, the first line in the file shall be the current position.
-s	Replace consecutive empty lines with a single empty line.
-t *tag*	Display the file containing the tag named by the *tag* argument. Note that for this to work, the file **tags** must reside in the current directory (see the **ctags** utility).

-u Always display backspaces as control characters (e.g. as the two-character sequence **^H**) and leave CR/LF (**\r\n**) sequences alone.

By default, **more** makes special use of backspaces and CR/LF (**\r\n**) sequences. If a backspace appears next to an underscore character, the character is displayed as underlined text, provided the terminal type supports underlined text. If a backspace appears between two identical characters, the first character is displayed as bold text, provided the terminal type supports bold text display.

-x *tabstop* Set tabs at the positions specified by *tabstop*. The default is four spaces, unless the **POSIX_STRICT** environment variable is defined, in which case it is eight spaces.

file A pathname of an input file. If no *file* operands are specified, **more** uses the standard input. If a *file* operand is the dash character (**-**), the standard input is read at that point in the sequence.

Description:

The **more** utility lets you view text files one screenful at a time. The utility determines the number of lines that make a full screen by looking in the terminal database. However, the **LINES** environment variable can be used to override the value found in the database, and the **-n** option can be used to override the **LINES** variable.

If the standard output isn't a terminal device, the number of lines that make up a full screen of text is considered to be infinite. In a pipeline, all input files are copied to the standard output in their entirety. On terminals, **more** displays text one screen at a time.

The **more** command can be very useful when another utility prints more information to the standard output than can be displayed on a single screen. By piping the output to **more**, you can scroll through the displayed output at leisure. For example:

```
ls | more
```

pipes the output from the **ls** command to **more**, allowing you to scroll through the output.

The **more** command is a link to **less**, which behaves according to the command name it was invoked as.

Environment variables:

EDITOR Used to select an editor.

LINES Interpreted as a decimal integer value to be used as the number of lines in a screenful.

MORE Interpreted as a string containing options described in the Options section of this utility, preceded by hyphens and separated by blank characters as on the command line. Command-line options override those specified in the **MORE** variable. The **MORE** variable takes precedence over the **TERM** and **LINES** variables.

TERM Interpreted as the name of the terminal type.

POSIX_STRICT

Interpret options according to POSIX specifications.

Exit status:

0 Successful completion.

>0 An error occurred.

Author:

Mark Nudelman

© 2005, QNX Software Systems

See also:

cat, **ctags**, **less**

Syntax:

mount [-v] [-w *secs***] -p** *block_special_file*

mount [-v] [-w *secs***]** *block_special_file directory* **[***flags***]**

mount [-v] -t *type type_specific_options*

Options:

-p *block_special_file*

Read the partition table from the named block special file and mount all partitions found. The partitions will appear as block special files with the form:

*block_special_file***t***partition-type-num*

That is, if the block special file was named **/dev/hd0** and the QNX 4 partition (type 77) was found, the block special file **/dev/hd0t77** would be created.

-t *type* Mount the type of filesystem specified. The **mount** utility does not know about all types of filesystem. For types other than **qnx4**, **mount** will invoke another utility named **mount_***type*, with the remaining command-line parameters supplied. The interpretation of the remaining parameters is therefore up to that utility. If there are no at-signs (**@**) or colons (**:**) in the arguments, the default type is **qnx4**. If at-signs or colons are found, **mount** will default to type **nfs** and will invoke the **mount_nfs** utility (QNX TCP/IP runtime product) to perform the mount operation.

-v Be verbose; write diagnostic messages indicating actions performed to the standard output.

-w *secs* Wait up to the indicated number of seconds for the block special file to appear. This is useful at boot time for slow-resetting devices. Default: 60 seconds.

block_special_file directory [*flags*]

Mount the named block special file as a filesystem named *directory*. Optional flags modify the way **Fsys** will work with the filesystem. Flags are:

- **-r** Mount as a read-only filesystem.
- **-a** Make all metadata (system) updates asynchronous.
- **-e** Don't allow executables to load.
- **-g** Don't allow persistent pregrown files.
- **-s** All data (user) updates are synchronous.
- **-u** Don't honor setuid bits.

Description:

You use the **mount** utility to mount disk partitions as block special files, and to mount block special files as QNX 4 filesystems. When you're mounting a block special file as a filesystem, the location in the pathname space where the filesystem is mounted is called a *mount point*.

The **mount** utility can also be used as a common front-end for mounting other types of filesystems (e.g. DOS, ISO 9660, NFS). This is accomplished by passing a **-t** *type* option to **mount** which tells it to invoke another utility to perform the actual mount operation. (The secondary mount utility must be found in the current path, **$PATH**. The filename of the executable will be **mount_***type*.)

Since the **-t** *type* option requires that **mount** *exec()* into another program, you can't use this form of **mount** within a QNX boot image. If you need to mount a non-QNX filesystem from the processes in the boot image, you must use the specific mount command for the filesystem type desired (e.g. **mount_dos**, **mount_iso9660**).

Examples:

Mount all partitions found on hard disk 0, and mount the QNX 4 partition as the root (this line is commonly seen in OS build files):

```
mount -p /dev/hd0 /dev/hd0t77 /
```

Mount floppy disk 0 as the directory **/fd** and make it read-only:

```
mount /dev/fd0 /fd -r
```

Mount the ramdisk as the temporary directory:

```
mount /dev/ram /tmp
```

Mount the QNX 4 partition found on the second hard disk as the user directory:

```
mount /dev/hd1t77 /home
```

Mount the ISO 9660 filesystem present on the CD-ROM device **/dev/cd0** as **/cd0**:

```
mount -t iso9660 /dev/cd0 /cd0
```

Note that in the above case, **mount** would invoke the following utility:

```
mount_iso9660 /dev/cd0 /cd0
```

Exit status:

0 Successful completion.

>0 An error occurred.

© 2005, QNX Software Systems

See also:

buildqnx, **mount_cdfs**, **mount_dos**, **mount_iso9660**, **mount_smb**, **umount**

mount_cdfs

Mount CD-ROM filesystems (QNX)

Syntax:

mount_cdfs *CD-ROM-block_special_file directory*...

Options:

See **mount_iso9660**.

Description:

The **mount_cdfs** utility is a link to **mount_iso9660**. It is used by the **mount** utility to mount CD-ROM block special files as filesystems. The **mount_cdfs** link exists so that users of the **mount** utility can specify the more intuitive **-t cdfs** to the **mount** command instead of **-t iso9660**.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

Iso9660fsys, **mount**, **mount_iso9660**, **mount_smb**

mount_dos

© 2005, QNX Software Systems

Mount DOS filesystems (QNX)

Syntax:

mount_dos *DOS_block_special_file directory* **...**

Options:

DOS_block_special_file

The DOS block special file (e.g. **/dev/fd0**) to be mounted.

directory Name of the resulting filesystem (e.g. **/dos/a**).

Description:

The **mount_dos** utility is used by the **mount** utility to mount DOS block special files as filesystems. It may also be run directly.

To succeed, **mount_dos** requires that the **Fatfsys** filesystem manager already be running.

Examples:

Typically, **mount_dos** is invoked via the **mount** utility. But if this isn't possible (e.g. if the device is being mounted from within the boot image), then you may run the **mount_dos** utility directly.

Start the **Fatfsys** manager, then mount the DOS "C" drive on **/dev/hd0t6** as **/dos/c**:

```
Fatfsys &
mount_dos /dev/hd0t6 /dos/c
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

Fatfsys, **mount**, **mount_cdfs**, **mount_iso9660**, **mount_smb**

mount_iso9660

© 2005, QNX Software Systems

Mount ISO 9660 filesystems (QNX)

Syntax:

mount_iso9660 *CD-ROM_block_special_file directory* **...**

Options:

CD-ROM_block_special_file directory

Mount, as a filesystem named *directory*, the named CD-ROM block special file.

Description:

The **mount_iso9660** utility is used by the **mount** utility to mount CD-ROM block special files as filesystems. It may also be run directly.

To succeed, **mount_is9660** requires that the **Iso9660fsys** filesystem manager already be running with the **-s** option.

Examples:

Typically, **mount_iso9660** is invoked via the **mount** utility. Only if this is not possible (e.g. if the device is being mounted from within the boot image) should the **mount_iso9660** utility be run directly.

Start the **Iso9660fsys** manager, then mount the CD-ROM on **/dev/cd0** as **/cd0**:

```
Iso9660fsys -s &
mount_iso9660 /dev/cd0 /cd0
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

Iso9660fsys, **mount**, **mount_cdfs**, **mount_smb**

mount_smb

© 2005, QNX Software Systems

Mount SMB filesystems (QNX)

Syntax:

mount_smb [-h] [-m a|u|n] [-n *node***] [-p] [-s] [-u** *user***]**
//[*netbios-name***]:***IP-name***/***resource*
local-path
[*password***]**

Options:

-h Don't substitute username for **homes** share name. If mounting a share with the reserved name **homes** (samba), **SMBfsys** will substitute the username for **homes**, thus avoiding permission problems when different users are mounting the share name **homes**. The **-h** disables this substitution.

-m a|u|n Map file names returned by directory read operations. SMB servers typically return names in all uppercase letters, which makes using them from QNX a bit awkward. **SMBfsys** can translate the names to lowercase letters, as directed by the letter following the option:

- **a** Always map the name to lowercase letters.
- **u** Map the name to lowercase letters only if the original name does not have any lowercase letters in it.
- **n** Never map the name to lowercase letters.

The default is **-m u** unless **-s** is specified, in which case **-m n** is the default.

-n *node* Tell the SMBfsys process running on the specified node to perform the mount. The default is to tell the **SMBfsys** process running on the same node **mount_smb** is invoked from.

-p	Mount as a printer service. Take the *local_path* command-line parameter as a file to be opened and written to. Generally the server will take whatever output is given and send it to a printer.
-s	Treat file names as case sensitive, if the server supports it.
-u *user*	Perform the mount as the specified user ID or username. (Default: current user)
netbios-name	Use *netbios-name* as the name of the server when establishing the NetBIOS session. By default, **mount_smb** assumes that the NetBIOS name is identical to the first portion of the *IP-name*, i.e. the portion up to the first dot (**"."**).
IP-name	This is the TCP/IP host name or IP address of the SMB server.
resource	This is the SMB service name that you are requesting to mount. The service names are established by the SMB server so you'll have to check with the system administrator of the server to find out what they are.
local-path	This is the local directory where the mounted resources will appear.
password	If the SMB server has established a password for the service being requested, provide it here. If no password is provided, **SMBfsys** will use an empty password when attempting to mount the service.

Description:

The **mount_smb** command is used to locally mount remote filesystems through connections to SMB servers. Rather than use **mount_smb** directly, you should use

mount -t smb *mount_smb_command_line*

You do not have to be **root** to use **mount_smb**. You do, however, require write permission on the directory where *local_path* is situated. That is, if *local_path* is **/smb/server1**, you need write permission on **/smb** to perform the mount (or **/** if **/smb** does not exist).

As mentioned in the **SMBfsys** documentation, a machine has two names — the NetBIOS name and the DNS name (the DNS name is what will be returned by the *gethostbyname()* library function). The **mount_smb** utility supports either type of lookup to determine the IP address of the server. If *server_name* contains no dots, and *netbios_name* is not explicitly specified (no colon in the command), then **mount_smb** first performs a NetBIOS name lookup on *server_name*. If that succeeds, the returned IP address is used as the IP of the server machine. If the NetBIOS name lookup fails to find anything, *server_name* is then looked up as a DNS name. If *netbios_name* is specified (colon on the command line) or there are dots in *server_name*, then *server_name* is either an explicit numeric IP address, or is subjected to a DNS name lookup only.

Examples:

Typically, **mount_smb** is invoked via the **mount** utility. Only if this is not possible should the **mount_smb** utility be run directly. The **mount -t smb** command will invoke **mount_smb** for you.

Start the **SMBfsys** manager, then mount a remote resource. The NetBIOS name of the server is APPSERVER, the server is found on the IP host 192.9.200.12 and the resource we want is **/docs** which was shared by APPSERVER. It will appear in the local filesystem as **/mnt/docs**:

```
SMBfsys &
mount_smb //APPSERVER:192.9.200.12/docs /mnt/docs
```

If 192.9.200.12 has an entry in the **/etc/hosts** file or you are running some form of name resolution, you could use the symbolic

name instead of the IP name. For instance, if 192.9.200.12 is the IP address for **marino**, then we could have used the following form :

```
mount_smb //APPSERVER:marino/docs /mnt/docs
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

user_smb, **SMBfsys**, **mount**, **mount_cdfs**, **mount_iso9660**

Mouse

© 2005, QNX Software Systems

Mouse manager (QNX)

Syntax:

Mouse [-n *name***] &**

Mouse [-n *name***] test**

Options:

-n *name*	The name to register (default is **qnx/mouse**) or the name of an installed driver to test.

Description:

The **Mouse** manager manages the mouse for system components which are not capable of dealing directly with the **Input** manager. This includes text-mode mouse applications (such as some mouse-aware editors and Watcom's **wd** debugger) and older versions of graphical user interfaces for QNX.

It reads and interprets a stream of data from the **Input** manager (through the file **/dev/mousein**), and processes this into an internal first-in first-out queue of significant mouse events. Application programs can "read" these mouse events by sending messages to the Mouse server via a set of mouse-specific functions.

To start or terminate **Mouse**, you must be logged in as the superuser (**root**).

Examples:

Start the mouse manager, auto-detecting the mouse type.

```
mousetrap start
```

Start the mouse manager, reading data through **/dev/mousein** from the **Input** driver which is handling a Microsoft compatible serial mouse on **/dev/ser1**:

```
Mouse &
Input -d/dev/mousein msoft fd -d/dev/ser1
```

Test that the mouse manager is functioning correctly:

```
Mouse test
```

Files:

For **Mouse &**: When run as a driver, **Mouse** will usually close both standard output and standard error after startup initializations have been completed. During initialization, warning/informative messages may be written to standard output. Any problem which will cause **Mouse** to terminate will cause a diagnostic message to be written to standard error.

The **Mouse** manager reads mouse data from the file **/dev/mousein** which is managed by **Input** (which has been given a **-d /dev/mousein** option).

For **Mouse test**:

in the test mode, **Mouse** will attempt to talk to the mouse driver attached to the filename specified by the **-n** or to **/dev/mouse**.

Test output will be written to standard output.

Exit status:

For **Mouse &** :

When run as a manager, **Mouse** will usually never terminate. However, it will terminate if errors are encountered during initialization or in response to being given bad command-line parameters. Also, **Mouse** will terminate if it has forked itself into the background (which it will attempt to do if it was not started in the background to begin with).

0 **Mouse** successfully forked a background copy of itself.

>0 An error occurred during initialization, or an attempt to fork a copy of itself into the background failed. A diagnostic message will have been written to the standard error.

For **Mouse test** :

0 User successfully terminated a test by pressing both left and right mouse buttons.

>0 User had to terminate the test by killing the **Mouse test** process, or **Mouse** was unable to start the test because the **Mouse** driver was not running.

Caveats:

If supplied with command-line parameters intended for earlier versions of **Mouse**, **Mouse** will automatically run **mousetrap** to auto-detect the mouse type and run **Input** appropriately. Note that **mousetrap** is actually a link to **inputtrap** which runs in a special mode when invoked as **mousetrap**.

See also:

Input, **inputtrap**, **mousetrap**

mouse_*()

Syntax:

Print command line necessary to start driver:

mousetrap [*options***] [query]**

Start driver:

mousetrap [*options***] start**

Options:

See **inputtrap**.

Description:

The **mousetrap** utility is a link to **inputtrap**, which is a general-purpose program to autodetect GUI input devices and start **Input** appropriately. When run as **mousetrap**, **inputtrap** will perform the additional function of starting the **Mouse** manager if it is not already running.

See **inputtrap** (Detect input devices and start Input Manager) for details on available options.

The **mousetrap** utility is used for mouse support only for text mode applications running on the console which use the mouse, for QNX Windows and for some older versions of the X window system for QNX. The **inputtrap** utility provides similar functionality for **Photon** and the X window system.

See also:

Input, **inputtrap**, **Mouse**

Mqueue

© 2005, QNX Software Systems

POSIX message queue server (QNX)

Syntax:

Mqueue [-anmrbd] [-s *message size*] **[-S** *read_buffer_size*] **[-x** *max messages*]

Options:

The following specify general behavior for the server:

-S *read buffer size*

Specifies the size of the read buffer. Messages smaller than this size will be read without an additional kernel call (defaults to 128).

The following specify which non-POSIX flags will be ORed into the *mq_flags* of a message queue if no structure is specified in the *mq_open()* call:

-a (**MQ_NOTIFY_ALWAYS**) notify even if queue contains messages when you call *mq_notify()*

-b (**MQ_PRIO_BOOST**) a 0 priority message will be boosted to the priority of the calling process

-d (**MQ_READBUF_DYNAMIC**) a *read()* (or *mq_receive()*) buffer may be smaller than the maximum message size.

-m (**MQ_MULT_NOTIFY**) more than one process can call *mq_notify()*

-n (**MQ_NONBLOCK**) *mq_send()*'s and *mq_receive()*'s won't block

-r (**MQ_PRIO_RESTRICT**) a process may not submit a message at a higher priority than itself

The following specify what values other fields of the **mq_attr** structure will default to if no structure is specified in the *mq_open()* call:

-s *message size*

The maximum size for any given message. If **MQ_READBUF_DYNAMIC** is not specified, this size also serves as a minimum.

-x *max messages*

The maximum number of messages on any message queue

Description:

The **Mqueue** manager is a POSIX 1003.1b message queue server which provides both the standard I/O API and the POSIX 1003.1b message queue API to the programmer. The command-line options allow the default attributes of a queue to be set very explicitly. In the event that a queue is to be created, and the new attributes have not been specified, the defaults specified on the command line will be used.

You must be logged in as **root** to start **Mqueue**.

See also:

mq_*() functions

msgprint

© 2005, QNX Software Systems

Display kernel event log (QNX)

Syntax:

msgprint [*file***]**

Options:

file Filename of the kernel event log which is to be displayed.

Description:

The **msgprint** utility displays kernel event logs in a readable ASCII format to standard output.

Examples:

Display the kernel event log contained in the file **default.mon**:

```
msgprint default.mon
```

See also:

monitor

Syntax:

mv [-f|-i] [-v|-V] *source_file target_file*

mv [-f|-i] [-v|-V] *source_file*... *target_dir*

Options:

-f	Force an overwrite; don't prompt for confirmation if the destination path exists. Overwrite even read-only files.
-i	Run interactively; write a prompt to the standard error output before moving any file that would overwrite an existing file. If confirmation is received, overwrite the existing file. Otherwise, go on to the next file.
-v	(QNX extension) Be verbose.
-V	(QNX extension) Be very verbose.
source_file	The pathname of a file or directory to be moved.
target_file	The new pathname of the file or directory to be moved.
target_dir	The pathname of an existing directory that the source file is to move to.

Description:

First syntax form:

mv [**-f|-i**] [**-v|-V**] *source_file target_file*

The **mv** utility moves the file named by the *source_file* operand to the destination specified by the *target_file* operand. This first syntax form is assumed when the final operand you specify doesn't name an existing directory.

Second syntax form:

mv [**-f|-i**] [**-v|-V**] *source_file... target_dir*

For each *source_file*, the **mv** utility moves the file to a destination file in the directory named by the *target_dir* operand. The destination's filename under the target directory will be the same as its basename (final path component).

For example:

```
mv dir/dir/myfile /existingdir
```

will move **dir/dir/myfile** to **existingdir/myfile**.

This second syntax form is assumed when either the destination names an existing directory, or when more than one source file is specified.

By default, **mv** overwrites an existing file without warning or confirmation whenever you have write permission on the file.

The **mv** utility asks you for confirmation if the following conditions are met:

- you haven't specified the **-f** option
- you lack write permission
- the standard input is a terminal

Upon receiving confirmation, **mv** overwrites the target file. It can do this only if the user owns the file or is the superuser.

If you want **mv** to request confirmation before overwriting any file, specify the **-i** (interactive) option. If you want **mv** to overwrite whenever possible without asking for confirmation, specify the **-f** (force) option.

Examples:

In the current directory, rename the file **orange** to **banana**.

```
mv orange banana
```

Files:

As long as the input files specified by each *source_file* are on the same device as the target, the *source_file* operand can be of any file type. If the source and target reside on different devices, the *source_file* will be copied to the target and then removed. If the *source_file* is a directory, this means that any FIFO or character special files under the original directory will not be copied. Since the copy will not be 100% successful, the original *source_file* will not be removed.

Exit status:

0 All input files were moved successfully.

>0 An error occurred.

Errors:

If the copying of a directory is prematurely terminated by a signal or error, **mv** may leave a partial copy of the directory at the destination. In this case, the directory tree at *source_file* will not have been modified.

Caveats:

When the *source_file* and *target_file* are on different filesystems (i.e. not on the same mounted partition), the **mv** utility will spawn the **cp** utility to copy the file(s), and if the **cp** succeeds, will spawn the **rm** utility to remove the originals.

See also:

cp, **pax**, **rm**

QNX 4 Utilities Reference — N to Z

© 2005, QNX Software Systems

This volume (*Utilities Reference — N to Z*) continues where the first volume leaves off, containing documentation for the utilities **nameloc** to **zcat**.

The first volume (*Utilities Reference — A to M*) covers the utilities **alive** to **mv**.

nameloc

Name locator (QNX)

Syntax:

nameloc [-e *node***] [-k** *node* **[-w]] [-p** *seconds***]**
[-S] [-s *node***] [-t** *seconds***]**

Options:

-e *node*	End node to poll (default is number of nodes).
-k *node*	Kick nameloc on the specified *node*, wait for it to talk to our node, then terminate. This is used on machines which boot from their own local hard disks but use the software licenses stored on a remote node.
-p *seconds*	Poll period (default is 1 second).
-s *node*	Start node to poll (default is 1).
-S	Slow poll — don't speed through "down" nodes.
-t *seconds*	Timeout on slow/hung nodes (default is 5 seconds).
-v	Be verbose; print out nodes that are being polled.
-w	Used only in conjunction with **-k**, causes nameloc to exit after kicking the nameloc on the remote node, instead of waiting for it to talk to our node.

Description:

The **nameloc** utility runs in the background and provides a network-wide naming service for all processes running under the operating system. You may run **nameloc** on up to 10 different nodes in the network — this provides redundancy in case a node running **nameloc** fails. Note, however, that running **nameloc** on just 3 nodes is usually more than sufficient. Each **nameloc** generates background network traffic, so the numbers of **nameloc**s should be limited.

A process may register one or more symbolic names. These names may be local to the node on which the process is running or global to the entire network. Global names all start with a slash. For example:

This name:	Is a:
qnx/fsys	local name
company/xyz	local name
/company/xyz	global name

The convention of prefixing a name with a company or product name is encouraged to reduce name conflicts between different vendors. The name locator is *not* required to locate names on the same node as the process performing the lookup. It is required if you wish to locate global names on other nodes.

Upon starting, **nameloc** will immediately poll each node in the network for its list of global names. It will then go into a slow poll mode to refresh this information. This slow poll period may be changed using the **-p** option. The slow polling is not the major means by which **nameloc** keeps informed of process global names in the network. Each time a name is registered or removed, all name locators are immediately informed. The slow poll is a safety net to handle extraordinary error conditions that might cause it to miss an attach or detach request.

When polled by a name locator, each node in the network remembers the polling node. It will remember up to the last 10 different nodes it was polled from. If it fails to receive a poll after a 10-minute period, it will forget that node and thus that name locator.

Examples:

Start **nameloc** with defaults:

```
nameloc &
```

Start **nameloc** with a 5 second poll period:

```
nameloc -p 5 &
```

Start **nameloc**, polling nodes 1 through 9 only:

```
nameloc -s 1 -e 9 &
```

Caveats:

Be careful to run **nameloc** only on those nodes with full network access. One of the functions of **nameloc** is to keep track of the *qnx_net_alive()* list, which is used by many utilities to quickly determine what nodes on the network are alive. If **nameloc** is run on a node which has less network access than do *any* of the nodes which can access it, those nodes may run into a situation where they believe they can talk to only a fraction of the nodes which are up. In addition, the list of global names can be similarly flawed (an incomplete list present), and licensing information can be wrong. For these reasons, we suggest that **nameloc** be run only on servers which can communicate with every node on the network.

If a node is having difficulty with its **alive** list, global names (see **sin gnames**) or licensing (see **licinfo**), use **sin info** to list the namelocators in use. Chances are good that the first name locator node in the list is at fault.

See also:

sin

Net

© 2005, QNX Software Systems

Network Manager (QNX)

Syntax:

Net [*options***]... &**

Options:

-A Never automatically add network mappings for previously unmapped machines which attempt to communicate with this node. (Default is to automatically learn new network mappings when previously unknown nodes initiate communication with this node.)

-b *brk_buf_numk* The amount of RAM (in kbytes) to be allocated for bridge packet buffers. (Default is 64) This memory won't actually be allocated until the first brouted packet arrives at this node.

-C Don't maintain "**netmap**" node/driver transmit counts

-d *num_drivers* The maximum number of network drivers that may be registered with **Net** at any one time (default is 2).

-e *num_log_elements*

The number of elements in the runtime error log used by the Network Manager and the network drivers (default is 200).

-E *mask* Inhibit **netinfo** logging messages as follows:

- **-E 1** filters out "unrecognized raw protocol" message.
- **-E 2** filters out "netmap change" message.
- **-E 3** filters out both.

-L *name* Choose name to attach to. Default is **qnx/net**.

-m *mapping* A logical-to-physical node ID mapping that is preloaded by Net. The mapping contains three fields: the first is a logical node ID; the second, a logical network ID; and the third, a physical network card ID (address). Multiple **-m** options are permitted. Note that a node always knows its own logical node ID (Process Manager **-l** ("el") parameter), logical network ID (network driver **-l** ("el") parameter), and physical network ID (obtained by a network driver from the hardware).

-n *num_nodemap*

The number of nodemap elements (default is the number of nodes in the OS license). These are used to cache logical-to-physical node mapping information. One element is required per remote node being communicated with at any time.

-q *num_qpkts* The number of elements in the Process Manager to Network Manager nonblocking queue (default is 2× the maximum number of processes specified to **Proc32**). You should need to use this option only in the unlikely event that the Process Manager runs out of elements. The more virtual circuits a node will have, the more elements it will require.

-r *num_raw* The number of raw protocols that may be registered with **Net** for reception of raw data by processes (default is 16).

-R Retry replies until connection is gone. Default is to not attempt to retry replies.

-S Allow raw apps to take over previously registered protocols. For example, you could temporarily usurp TCP/IP, then relinquish it.

-t *tx_fail_time* The number of seconds a network driver will be "avoided" by the Network Manager if the driver

fails to transmit to a particular node (default is 40). This is relevant only when multiple networks are being used.

-T Don't log runtime errors to the trace log (**traceinfo**); use the built-in **netinfo** log only.

-v Be verbose. If a runtime error occurs in **Net** or in any network driver, a message with an error code will be displayed on the console. This option is normally used only for debugging.

Description:

The **Net** manager provides services that the Process Manager and Kernel require to provide transparent networking. It does this via network drivers. The Network Manager is typically built into an operating system image or started in the **sysinit** file; it is rarely started from the command line. If you experience any problems, you should run the **netinfo** utility to get information about recent significant network events and errors.

CAUTION: Don't change the priority of **Net** or of a **Net.*** driver — unexpected behavior may result if you do!

The versions of **Net** in QNX 4.23 and later support "*brouting*," a QNX-to-QNX bridge/routing scheme which allows QNX nodes on different physical IEEE 802-based networks to talk to one another through a common node or nodes which are connected to both networks. This scheme is described in detail in the technical note **/etc/readme/technotes/Net_brouting** available on QUICS.

Brouting is automatic if the network maps (see **netmap**) are properly configured on all nodes. To work, the following conditions must be met:

- each physical network present has been assigned a unique network number for use in the **netmap** table. (e.g. you can't have a net 1 bridged to another net 2 which is in turn bridged to a second net 1)

- each network node on any of the physical networks has a unique node number (e.g. you can't have a node 1 on net 1 and a node 1 on net 2)
- each node has a complete netmap table which declares all the nodes on all the networks, even if that particular node does not have a direct connection to all the physical networks

There is additional mention of the network mapping requirements for brouting in the documentation for the **netmap** utility.

Examples:

Start the Network Manager, with defaults:

```
Net &
```

Support a node which will have 3 network cards installed:

```
Net -d 3 &
```

Run net with an increased error log size:

```
Net -e2000 &
```

Don't log trace events:

```
Net -T &
```

Support a node which must boot up already knowing the network mapping to node 1, since it is booting across the network using a boot ROM and needs to be able to talk to node 1 to obtain the system initialization file and binaries necessary to complete its startup:

```
Net -m "1  1  0000C0 469330" &
```

This example lets **Net** preload its mapping table with the information that logical node ID 1 may be reached via logical network 1 (which must be the same as the network driver's) by transmitting to physical

address 0000C0 469330. This allows the node to execute the **sysinit.***node* file on node 1, which contains the **netmap** utility. The **netmap** utility will open a node mapping file located on the server (node 1) and then tell **Net** about all of the other logical-to-physical node mappings found in this mapping file.

The physical address in the above example is a 48-bit IEEE 802 address. Some network hardware supports less than 48 bits. For example, in the case of Arcnet, the physical address length supported is only 8 bits, so the third field would consist of only one byte. By default, the physical address is assumed to be in hex, but for Arcnet you can specify an 8-bit address in decimal if you prefix the address with a **t** (e.g. **t1**). For more information, see the **netmap** utility.

Signals:

The **Net** manager ignores SIGPWR.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

Net.*, **netboot**, **netinfo**, **netmap**, **netpoll**, **traceinfo**, **tracelogger**

Syntax:

Net.arc20 [-i *irq***] [-l** *log_net_id***] [-n** *phys_nid***]**
[-p *io_port***] [-r** *media_rate***] [-s] [-v] &**

Options:

-i *irq*	Use this interrupt. The default is 2. If you use an interrupt other than 2, you must use the -i option
-l *log_net_id*	("el") Connect to this logical network (default is 1).
-n *phys_nid*	Program the Arcnet adapter to be this physical node ID. Without the -n option, the I/O port at the (base+9) will be read and that value will be used.
-p *io_port*	Use this I/O port base, specified as a hexadecimal number *without* a leading **0x** (default is **2E0**). If you use another I/O port, you must use the **-p** option.
-r *media_rate*	Advertise this bit-transmission rate (default is 2500000). The Network Manager (**Net**) uses this information when multiple network drivers are running.
-s	Terminate the driver if SIGPWR is delivered as a result of a shutdown command. By default, the driver doesn't terminate after a shutdown command; this behavior helps ensure an orderly termination of the virtual circuits.
-v	Be verbose: print out hardware info on the console when starting up.

Description:

The **Net.arc20** network driver communicates directly with networking hardware based on the SMC COM20020 Arcnet chipset. The driver provides the Network Manager (**Net**) with reliable data transfer over an Arcnet network.

To install multiple network cards in one machine you should specify a unique logical network ID with the **-l** option to every network driver option to every invocation of **Net.arc20**

Examples:

Start the **Net.arc20** driver for a COM 20020 chip at I/O port base 0x2E0 using interrupt 2:

```
Net.arc20 &
```

Start the **Net.arc20** driver for a COM 20020 chip at I/O port base 0x300 using interrupt 2:

```
Net.arc20 -p300 &
```

Start the **Net.arc20** driver for a COM 20020 chip at I/O port base 0x320 using interrupt 5:

```
Net.arc20 -p320 -i5 &
```

Files:

Net.arc20 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

This drivers ignores SIGPWR unless the **-s** option is specified.

Exit status:

Net.arc20 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.arc20** has shut down successfully and cleanly de-registered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **shutdown**

Installation & Configuration

Net.arcnet

© 2005, QNX Software Systems

Arcnet network driver (QNX)

Syntax:

Net.arcnet [*options***] &**

Options:

-b	Disable reception of broadcast packets.
-i *irq*	Use the specified interrupt line (IRQ). This option overrides automatic IRQ detection.
-l *log_net_id*	("el") Connect to the specified logical network ID. (Default: 1)
-m *mem*	Use the specified memory segment (hexadecimal). This option overrides automatic RAM detection.
-n *num_retries*	Specify the number of retries after a transmit failure before giving up. (Default: 50)
-p *ioport*	Specify the I/O port location for a standard SMC Arcnet card, specified as a hexadecimal number *without* a leading **0x**.
-r *media_rate*	Advertise the specified bit-transmission rate. The default is 2500000. This information is used by the Network Manager when multiple drivers are running.
-s	Terminate on SIGPWR. (Default: stay running)
-t *ticks*	Specify the number of 50 millisecond ticks before primary timeout (Default: 40)
-T *ticks*	Specify the number of 50 millisecond ticks before secondary timeout (Default: 40)
-v	Be verbose; print out hardware info on the console when starting up.

Description:

The **Net.arcnet** network driver communicates directly with either a Corman Arcnet card with memory mapped registers, or a standard Arcnet card with I/O mapped registers. It provides the Network Manager with reliable data transfer over an Arcnet network.

By default, **Net.arcnet** will detect all hardware parameters automatically. However, you can specify any or all of these parameters by using the options described above.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.arcnet** file.

If you install multiple Arcnet cards, you must specify a unique logical network ID (**-l** option) for every driver. On an ISA bus computer (e.g. AT), you must also specify a unique memory segment (**-m** *mem* option) for every Corman card. For a standard Arcnet card, you must specify **-p** *ioport*.

Examples:

One Corman or standard Arcnet card is installed. Use automatic IRQ and RAM detection:

```
Net.arcnet &
```

Use IRQ 3 and set the memory segment to be located at DC00 hex:

```
Net.arcnet -i 3 -m DC00 &
```

Two Corman Arcnet cards are installed at memory segments DC00 and CC00; use automatic IRQ detection for both:

```
Net.arcnet -l 1  -m DC00 &
Net.arcnet -l 2  -m CC00 &
```

Two standard Arcnet cards are installed at I/O ports 300 and 260:

```
Net.arcnet -l 1 -p 300 &
Net.arcnet -l 2 -p 260 &
```

© 2005, QNX Software Systems

Files:

Net.arcnet closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.arcnet** driver ignores SIGPWR unless **-s** is specified.

Exit status:

Net.arcnet terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.arcnet** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo** *Installation & Configuration*

Syntax:

Net.arcxir -n *phys_nid* **[-i** *irq***] [-l** *log_net_id***]**
[-p *io_port***] [-r** *media_rate***] [-s] [-v] &**

Options:

-i *irq*
Use this interrupt line with the parallel port. By default, interrupts aren't used. The first and second parallel ports are typically associated with interrupts 7 and 5, respectively.

-l *log_net_id*
("el") Connect to this logical network (default is 1).

-n *phys_nid*
Program the Xircom adapter to be this physical node ID. You *must* specify this node ID for the driver to function, and the ID must be unique for the Arcnet network to function.

-p *io_port*
Use this I/O port base, specified as a hexadecimal number *without* a leading **0x** (default is 378, the address of the first parallel port on most machines). If the default value fails, you can try 278, usually the address of the second parallel port.

If your parallel port is on a TTL monochrome video card, the address may be 3BC.

-r *media_rate*
Advertise this bit-transmission rate (default is 2500000). The Network Manager (**Net**) uses this information only when multiple network drivers are running.

-s
Terminate the driver if SIGPWR is set as a result of a shutdown command. By default, the driver doesn't terminate after a shutdown command; this behavior helps ensure an orderly termination of the virtual circuits.

-v Be verbose: print out hardware info on the console when starting up.

Description:

The **Net.arcxir** network driver communicates directly with a Xircom Pocket Arcnet Adapter II, which plugs into a parallel port of your computer. The driver provides the Network Manager (**Net**) with reliable data transfer over an Arcnet network.

You *must* specify a physical node ID with the **-n** option since **Net.arcxir** doesn't use the parameters programmed by Xircom's DOS software.

By default, **Net.arcxir** assumes that the Xircom adapter is connected to the first parallel port and that interrupts aren't used. The driver conservatively polls the parallel port, which always works.

If your parallel port hardware supports interrupts (and most do, but may not have the jumpers configured to enable them), you can increase performance by specifying the **-i** option.

Usually the first parallel port drives IRQ 7 and the second parallel port drives IRQ 5, but check your computer's documentation and parallel port hardware's jumper configuration to be certain.

To install:	You should:
multiple network cards in one machine	specify a unique logical network ID with the **-l** option to every network driver
multiple Xircom cards in one machine	specify a unique I/O address with the **-p** option to every invocation of **Net.arcxir**

Examples:

Use the first (and probably the only) parallel port; don't use interrupts, and program the adapter to be physical node 1:

```
Net.arcxir -n 1 &
```

Use the first parallel port and IRQ 7 (for better performance), programming the adapter to be physical node 12:

```
Net.arcxir -n 12 -i 7 &
```

Use the second parallel port and polling instead of interrupts; program the adapter to be physical node 3:

```
Net.arcxir -n 3 -p 278  &
```

Use the second parallel port and IRQ 5; program the adapter to be physical node 2:

```
Net.arcxir -n 2 -p 278 -i 5 &
```

Files:

Net.arcxir closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.arcxir** driver ignores SIGPWR unless the **-s** option is specified.

© 2005, QNX Software Systems

Exit status:

Net.arcxir terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.arcxir** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **shutdown**

Installation & Configuration

Net.crys8900

Crystal 8900/8920 Ethernet network driver(QNX)

Syntax:

Net.crys8900 [-A] [-d *chnl***] [-F] [-i** *irq***]**
[-m *mac***] [-l** *log_net_id***]**
[-n *tx_num_retries***] [-P] [-p** *io_port***]**
[-r *media_rate***] [-t** *tx_retry_ticks***] [-v] &**

Options:

-a *io_port*
: This option is no longer valid in QNX 4.25 and later. Use **-p** instead.

-A
: Use AUI.

-d *chnl*
: Specify DMA channel.

-F
: Enable full-duplex operation.

-i *irq*
: Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect doesn't work on an ISA card, you have a hardware conflict.

-l *log_net_id*
: ("el") Connect to the specified logical network. The default is 1.

-m *mac*
: Specify MAC address.

-n *tx_num_retries*
: After failing to transmit to a remote node, retry transmission no more than this number of times. The default is 7.

-P
: Program the NIC for promiscuous **rx** mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

-p *io_port*
: Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default (**0x300**).

-r *media_rate*	Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager when multiple network drivers are running.
-t *tx_retry_ticks*	The number of 50-millisecond intervals between **tx** retries. The default is 6.
-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

Since the release of QNX 4.25, the **Net.crys8900** network driver replaces the **Net.ether8900** driver.

The **Net.crys8900** network driver communicates directly with a Crystal 89xx-compatible Ethernet network card. It provides the Network Manager with reliable data transfer over an Ethernet network. Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.crys8900** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one Crystal 89xx-compatible network card, you must also specify the **-p** and **-i** options for every invocation of **Net.crys8900**.

Examples:

On a Crystal 89xx autodetect adapter:

```
Net.crys8900  &
```

On a Crystal 89xx, use I/O port 300h and IRQ 5:

```
Net.crys8900 -p300 -i 5  &
```

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.epic

© 2005, QNX Software Systems

SMC 83C170 Ethernet network driver (QNX)

Syntax:

Net.epic [-B] [-f *tx_forget_time***] [-F]**
[-I *pci_index***] [-i** *irq***]**
[-l *log_net_id***] [-M]**
[-n *tx_num_retries***] [-o] [-P]**
[-p *io_port***] [-r** *media_rate***]**
[-s *speed***] [-t** *tx_retry_ticks***] [-v] &**

Options:

-a *io_port*
This option is no longer valid in QNX 4.25 and later. Use **-p** instead.

-B
Use the BNC (coax) port.

-f *tx_forget_time*
After a driver-level NACK is received from a remote node, **Net.epic** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 2 seconds, after which the eternally optimistic **Net.epic** stops throttling its output. The **-f** option allows you to change the default 2 seconds above.

-F
Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-I *pci_index*
Have the driver look at this specific PCI Index for an epic-compatible card. You should have to specify **-I** only if you have more than one epic-compatible card installed (e.g. **-I0**, **-I1**, **-I2**).

-i *irq*
Program the card to use the specified hardware interrupt. This option overrides the default

autodetect. If the autodetect doesn't work on an ISA card, you have a hardware conflict. This option should rarely be used.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 7.

-o Use the fiber-optic port.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address. In some earlier versions of QNX, the **-p** option was used to specify the PCI index. Use the **-I** option to specify the PCI Index of the card being used.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-s *speed* Set the speed at which the card should run. Only possible with 100Mbit cards. (e.g. **-s10** or

	-s100) This disables the auto-negotiation capability of Ethernet card.
-t *tx_retry_ticks*	The number of 50 millisecond intervals between transmit retries. The default is 6.
-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.epic** network driver communicates with SMC's 83C170 PCI Ethernet adapter to provide the Network Manager with reliable data transfer over an Ethernet network. The driver supports PHY auto-negotiation with auto-negotiating partners. Auto-negotiation can be disabled by using the **-s** (speed) command-line option. Full-duplex operation can also be forced, using the **-F** option.

The driver lets you select a card's coax port using the **-B** option or a fiber-optic port using the **-o** option (for cards with LXT970 or Altima Phy interfaces).

Most of the time, you shouldn't need to specify the I/O port or the hardware interrupt to **Net.epic** — it should autodetect these parameters from the PCI BIOS.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.epic** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one epic-compatible PCI network card, you must also specify the **-I** option for every invocation of **Net.epic**. (Use **-I0** for the first card, **-I1** for the second, etc.)

Examples:

Start the **Net.epic** driver with default parameters:

```
Net.epic &
```

Start the **Net.epic** driver and force the speed to 100Mbit:

```
Net.epic -s100  &
```

Start two copies of the driver running on separate cards; one on logical network 1, the other on logical network 2:

```
Net.epic -I0 -l1 &
Net.epic -I1 -l2 &
```

Files:

Net.epic closes its standard input, standard output, and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.epic** driver ignores SIGPWR.

Exit status:

Net.epic terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.epic** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.ether1000

© 2005, QNX Software Systems

NE1000/2000 Ethernet network driver (QNX)

Syntax:

Net.ether1000 **[-6] [-a] [-D** *id***] [-f** *forget_time***]**
[-h *hang_ticks***] [-i** *irq***] [-I** *index***]**
[-l *log_net_id***] [-M] [-n** *tx_num_entries***]**
[-p *io_port***] [-P] [-r** *media_rate***]**
[-R] [-s *ram_size***] [-t** *tx_retry_ticks***]**
[-v] [-V *id***] &**

Options:

-6 Instruct the driver that a 16-bit NE2000 card with 16K RAM is installed. This option overrides the driver's automatic detection of the NE2000 card, and should rarely be needed.

-a Use a 16-bit AT/LANTIC card run in 8-bit mode.

-D *id* Specify device ID.

-f *forget_time* After a driver-level NACK is received from a remote node, **Net.ether1000** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which the driver stops throttling its output. **-f** allows you to change the default 10 seconds above.

-h *hang_ticks* Specify the number of 50ms ticks to wait until the driver concludes that the chip is hung and resets it. Default: 10 (500ms).

-i *irq* Use this interrupt line (IRQ). This option overrides the automatic IRQ detection the driver performs if a boot ROM is installed and should rarely be needed.

-I *index* Specify the PCI index of the controller in the machine, where *index* is a value between 0 and the number of adapters.

-l *log_net_id*	("el") Connect to this logical network (default is 1).
-M	Enable reception of multicast packets.
-n *tx_num_retries*	After failing to transmit to a remote node, retry transmission no more than this number of times (default is 3).
-p *io_port*	Use this base I/O port address, specified as a hexadecimal number *without* a leading **0x**. This option overrides the automatic port detection the driver performs.
-P	Run promiscuously: buffer every packet, whether or not it's destined for this node. (A node usually discards any packet not addressed to it.) You may use this option in conjunction with **netsniff** to monitor other traffic on the network.
-r *media_rate*	Advertise this bit-transmission rate (default is 10000000). The Network Manager uses this information only when multiple drivers are running.
-R	Perform time-consuming data integrity checks. Use only with PCMCIA cards.
-s *ram_size*	Set the shared RAM segment on the card to this length (given in kilobytes). This option overrides the default of 8K for the NE1000 and 16K for the NE2000, and should rarely be needed.
-t *tx_retry_ticks*	Wait this many 50-millisecond intervals between transmit retries (default is 20).
-v	Be verbose: print out hardware info on the console when starting up.
-V *id*	Specify vendor ID.

© 2005, QNX Software Systems

Description:

The **Net.ether1000** network driver communicates directly with a Novell NE1000 (8-bit) or NE2000 (16-bit) Ethernet network card; it also works with compatible cards. The driver provides the Network Manager (**Net**) with reliable data transfer over an Ethernet network.

The driver autodetects the following PCI NE2000 clones:

- Realtek 8029
- Winbond 89C940
- Compex RL2000
- Netvin NV5000 SC
- Via 82C926

CAUTION: Installing a card at I/O port 360 may result in a conflict with the parallel port!

If you have installed a boot ROM you must use one of the following Novell configurations; **Net.ether1000** will automatically detect any IRQ.

Novell option	ROM address	I/O port	Default boot from:
0	C800	300	network
2	D000	340	disk
9	CC00	320	network
X	D400	360	network

Before installing a boot ROM or network card, note the following:

To install:	You should:
a boot ROM	read the documentation that came with the boot ROM
an NE1000 or NE2000 card	read the technote contained in the file **Net.ether1000** in **/etc/readme/technotes**.
multiple network cards in one machine	specify a unique logical network ID (**-l** option) to every network driver
multiple NE1000 or NE2000 or compatible cards in one machine	specify **-p** to every invocation of **Net.ether1000** even if a boot ROM is installed in one or more of the cards

Examples:

With a boot ROM installed...

Start the driving using Novell option 9 (IRQ 5, I/O port 0x320, ROM at 0xCC00):

```
Net.ether1000 &
```

Support two cards — one using Novell option 4 (IRQ 2, I/O port 0x300) on logical network 1, and the other using Novell option 9 (IRQ 5, I/O port 0x320, ROM at 0xCC00) on logical network 2:

```
Net.ether1000 -p 300 &
Net.ether1000 -p 320 -l 2 &
```

With *no* boot ROM installed...

Start the driver using IRQ 10 and I/O port 0x300:

```
Net.ether1000 &
```

Start the driver using IRQ 5 and I/O port 0x320:

```
Net.ether1000 &
```

Start the driver using IRQ 2 and I/O port 0x340:

```
Net.ether1000 &
```

Files:

Net.ether1000 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether1000** driver ignores SIGPWR.

Exit status:

Net.ether1000 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0	**Net.ether1000** has shut down successfully and cleanly deregistered from **Net**.
>0	An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Syntax:

Net.ether2100 **[-b] [-d** *dma_channel***] [-i** *irq***]**
[-f *tx_forget_time***] [-l** *log_net_id***]**
[-m *mem_cycle***] [-M] [-n** *tx_num_retries***]**
[-o] [-p *io_port***] [-r** *media_rate***]**
[-R *shared_mem_addr*,*shared_mem_size***]**
[-s *speed***] [-I** *pci_index***] [-t** *tx_retry_ticks***]**
[-v] [-P] &

Options:

-b Enable linear burst mode, on VL-bus and PCI cards only. This option has no effect on ISA bus cards. Note: this does not appear to work on most cards, and is disabled by default.

-d *dma_channel* Use the specified DMA channel. The default is 5. Other choices include 6 and 7; DMA channel 3 will not function correctly. Note that a DMA channel is required only for ISA cards. VL-bus and PCI cards do not require a DMA channel, and it is not enabled by default for these cards.

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.ether2100** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which the eternally optimistic **Net.ether2100** stops throttling its output. The **-f** option allows you to change the default 10 seconds above.

-i *irq* Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect does not work on an

ISA card, you have a hardware conflict. This option should rarely be used.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-m *mem_cycle* Number of nanoseconds per 16-bit bus master memory read/write cycle, ISA bus only. This option has no effect on VL-bus or PCI cards.

The default is 350, which is equivalent to the 5.7MB/sec speed of the bus-mastering Adaptec SCSI controller.

Valid cycle times are multiples of fifty. If you have problems, you might try 400, which is equivalent to 5.0 MB/sec. If you are courageous and have fast hardware, you could experiment with faster cycle times (e.g. 300, 250). Note that the lower and upper bounds are 150 and 850 respectively.

-M Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 3.

-o Use fiber optic links.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default autodetect, which scans ISA I/O ports 300, 320, 340, and 360, in that order, and also checks for PCI cards.

If you have two NE1500T or NE2100 cards installed, you must specifically tell each driver which card to talk to with the **-p** option.

If you also have an NE1000 or NE2000 card installed, you should definitely use the **-p** option to **Net.ether2100** as the NE1000/2000 cards do not enjoy being probed by the ISA autodetect code of the **Net.ether2100**.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 10000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-R *shared_mem_addr*,*shared_mem_size*

Set the address and size of the shared memory segment used by the AMD PCnet ISA II (79C961) ethernet card. The default is to read these values out of the EEPROM.

-S *pci_index* This option is no longer valid in QNX 4.25 and later. Use **-I** instead.

-s *speed* Set the speed at which the card should run (100Mbit cards only). e.g. **-s10** or **-s100** Use of this option disables the auto-negotiation capability of ethernet card.

-I *pci_index* Have the driver look at this specific *pci_index* for the card. You should have to specify **-I** only if you have more than one card installed. (e.g. **-I0**, **-I1**, **-I2**)

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries. The default is 20.

-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.ether2100** network driver communicates directly with a Novell NE1500T or NE2100 or compatible Ethernet network card which uses the Advanced Micro Devices (AMD) 79C960 and 79C961 single-chip Ethernet controllers for ISA, or 79C965 for VL-bus, or 79C970, 79C971, 79C972, 79C974 for PCI.

Net.ether2100 provides the Network Manager with reliable data transfer over an Ethernet network.

Most of the time, you should not need to specify the I/O port or the hardware interrupt to **Net.ether2100** — it should autodetect these parameters. However, if you use a DMA channel other than 5, you must use the **-d** option.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether2100** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one NE1500T/NE2100 or compatible network card, you must also specify the **-p** options for every invocation of **Net.ether2100**.

Examples:

Start the **Net.ether2100** driver on either an ISA or PCI NE1500T or NE2100, using the autodetected I/O port and hardware interrupt, and DMA channel 5:

```
Net.ether2100 &
```

Start the driver on either an ISA NE1500T or NE2100, using the autodetected I/O port and hardware interrupt, and DMA channel 6:

```
Net.ether2100 -d 6 &
```

Start two **Net.ether2100** drivers for two ISA cards installed in one machine. One card is using I/O port 300h and IRQ 10 and DMA channel 5 on logical network 1. The second card is using I/O port 320 and IRQ 12 and DMA channel 6 on logical network 2.

```
Net.ether2100 -p 300 &
Net.ether2100 -p 320  -d 6  -l 2  &
```

Start two **Net.ether2100** drivers for two PCI cards installed in one machine. One card is PCI index 0, the second PCI index 1.

```
Net.ether2100 -I0 &
Net.ether2100 -I1 &
```

Files:

Net.ether2100 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether2100** driver ignores SIGPWR.

Exit status:

Net.ether2100 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether2100** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

© 2005, QNX Software Systems

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Description:

Since QNX 4.25, the **Net.ether21x4x** driver is obsolete and has been replaced by **Net.tulip**.

Net.ether503

© 2005, QNX Software Systems

3Com 503 Ethernet network driver (QNX)

Syntax:

Net.ether503 [*options***] &**

Options:

-a Use the card's AUI (15-pin) port instead of its onboard transceiver (BNC, TP).

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.ether503** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which **Net.ether503** stops throttling its output. **-f** allows you to change the default 10 seconds above.

-h *hang_ticks* Specify the number of 50ms ticks to wait until the driver concludes that the chip is hung and resets it. Default: 10 (500ms).

-i *irq* Use this interrupt line (IRQ). You can specify a value from 2 to 5; the default is 5.

-l *log_net_id* ("el") Connect to this logical network (default is 1).

-m *mem* Use this memory segment (hexadecimal number, *not* be preceded by **0x**). This option overrides the driver's automatic detection of where the shared RAM is located.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times (default is 3).

-p *io_port*	Use this I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the driver's automatic I/O port detection.
-P	Run promiscuously: buffer every packet, whether or not it's destined for this node. (A node usually discards any packet not addressed to it.) You may use this option in conjunction with **netsniff** to monitor other traffic on the network.
-r *media_rate*	Advertise this bit-transmission rate (default is 10000000). The Network Manager uses this information only when multiple network drivers are running.
-s *ram_size*	Set the shared RAM segment on the card to this length (default is 8K). The *ram_size* argument is given in kilobytes (decimal).
-t *tx_retry_ticks*	Wait this many 50-millisecond intervals between transmit retries (default is 20).
-v	Be verbose: print out hardware info on the console when starting up.

Description:

The **Net.ether503** network driver communicates directly with a now-obsolete 3Com 503 Ethernet network card operating in shared memory mode. The driver provides the Network Manager (**Net**) with reliable data transfer over an Ethernet network.

By default, **Net.ether503** automatically determines the I/O port base and the segment of the shared RAM. However, it doesn't automatically detect the hardware interrupt (IRQ) since the 3Com 503 card cannot store this parameter. **Net.ether503** will use IRQ 5 unless you specify a different IRQ with the **-i** option.

The 3Com 503 card supports only IRQs 2, 3, 4, and 5. Make sure no other devices are using the same interrupt as the 3Com 503 card.

Before installing your card, note the following:

To install:	You should:
a 3Com 503 card	read the technote contained in the file **Net.ether503** in **/etc/readme/technotes**.
multiple network cards in one machine	specify a unique logical network ID (**-l** option) to every network driver
multiple 3Com 503 network cards in one machine	specify the **-p** option to every invocation of **Net.ether503**

Examples:

Use the default IRQ as well as automatic detection of I/O port and shared RAM:

```
Net.ether503 &
```

Do the same as above, but use IRQ 2:

```
Net.ether503 -i 2 &
```

Use the default IRQ, an I/O port base of 0x250, and automatic detection of shared RAM (normally, you need to specify an I/O port only if you've installed multiple 3Com 503 cards — with multiple cards present, the driver doesn't automatically detect the I/O port):

```
Net.ether503 -p 250 &
```

Use the default IRQ, a RAM segment of 0xCC00, and automatic detection of the I/O port (you don't normally need to specify the RAM segment location since the driver can automatically determine this once it has determined the I/O port base):

```
Net.ether503 -m CC00 &
```

Use IRQ 2, an I/O port base of 0x310, and a RAM segment of 0xD800:

```
Net.ether503 -i 2 -p 310 -m D800 &
```

Support two cards — one using IRQ 2 and an I/O port base of 0x2A0 on logical network 1, the other using IRQ 5 and an I/O port base of 0x330 on logical network 2. Use automatic RAM segment detection for both:

```
Net.ether503 -i 2 -p 2A0 &
Net.ether503 -p 330 -l 2 &
```

For this setup, you could configure the RAM segment at 0xC800 for the first card and at 0xCC00 for the second card. To configure the RAM segment, see the card manufacturer's documentation.

Files:

Net.ether503 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether503** driver ignores SIGPWR.

Exit status:

Net.ether503 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether503** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

© 2005, QNX Software Systems

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.ether509

3Com 509 Ethernet network driver (QNX)

Syntax:

Net.ether509 [*options***] &**

Options:

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.ether509** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which **Net.ether509** stops throttling its output. **-f** allows you to change the default 10 seconds above.

-i *irq*
: Use this interrupt line (IRQ). The default is autodetect.

-l *log_net_id*
: ("el") Connect to this logical network (default is 1).

-M
: Program card to receive multicast packets. The default is not to.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times (default is 3).

-p *io_port*
: Use this I/O port base, specified as a hexadecimal number *without* a leading **0x**. You should need this option only when multiple 509 cards are installed.

-P
: Run promiscuously: buffer every packet, whether or not it's destined for this node. (A node usually discards any packet not addressed to it.)

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-r *media_rate* — Advertise this bit-transmission rate (default is 10000000). The Network Manager uses this information only when multiple network drivers are running.

-t *tx_retry_ticks* — Wait this many 50-millisecond intervals between transmit retries (default is 20).

-v — Be verbose: print out hardware info on the console when starting up.

Description:

The **Net.ether509** network driver communicates directly with a 3Com 509 Etherlink III Ethernet network card. The driver provides the Network Manager (**Net**) with reliable data transfer over an Ethernet network.

By default, **Net.ether509** automatically determines the I/O port base and the IRQ.

Before installing your card, note the following:

To install:	You should:
a 3Com 509 card	read the technote contained in the file **Net.ether509** in **/etc/readme/technotes**.
multiple network cards in one machine	specify a unique logical network ID (**-l** option) to every network driver
multiple 3Com 509 network cards in one machine	specify the **-p** option to every invocation of **Net.ether509**

Examples:

One 509 card is installed. Use automatic detection of I/O port and IRQ:

```
Net.ether509 &
```

Support two cards — one using IRQ 2 and an I/O port base of 0x220 on logical network 1, the other using IRQ 5 and an I/O port base of 0x300 on logical network 2:

```
Net.ether509 -p 220 &
Net.ether509 -p 300 -l 2 &
```

Files:

Net.ether509 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether509** driver ignores SIGPWR.

Exit status:

Net.ether509 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether509** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

© 2005, QNX Software Systems

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.ether589

3Com 3C589 PCMCIA Ethernet network driver (QNX)

Syntax:

Net.ether589 -i *irq* **-p** *io_port*
[-B] [-f *tx_forget_time***] [-l** *log_net_id***]**
[-M] [-n *tx_num_retries***]**
[-P] [-r *media_rate***]**
[-t *tx_retry_ticks***] [-v]**

Options:

-a *io_port*
This option is no longer valid in QNX 4.25 and later. Use **-p** instead.

-f *tx_forget_time*
After a driver-level NACK is received from a remote node, **Net.ether589** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 3 seconds, after which the eternally optimistic **Net.ether589** stops throttling its output. The **-f** option allows you to change the default 3 seconds above.

-B
Use BNC connector. (Default: 10BASE-T)

-i *irq*
Program the card to use the specified hardware interrupt. This option must be specified. The driver does not autodetect the irq.

-l *log_net_id*
("el") Connect to the specified logical network. The default is 1.

-M
Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 7.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option is required because the driver doesn't autodetect the port address.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 10000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries. The default is 6.

-v Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.ether589** network driver interacts with a 3Com 3C589 EtherLink III PCMCIA Ethernet card to provide the Network Manager with reliable data transfer over an Ethernet network.

This driver does not support autodetection of IO port or IRQ. For this reason, the **-p** *io_port* and **-i** *irq* options must always be specified.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether589** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

Examples:

Start the driver on io_port 0x300 with irq 5:

```
Net.ether589 -p300 -i5 &
```

Start the driver on logical network 2 with an io_port of 0x320 and an irq of 5:

```
Net.ether589 -p320 -i5 -l2 &
```

Files:

Net.ether589 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether589** driver ignores SIGPWR.

Exit status:

Net.ether589 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether589** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

© 2005, QNX Software Systems

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.ether7231

AT&T 7231 Ethernet network driver (QNX)

Syntax:

Net.ether7231 [*options***] &**

Options:

-a *address*
Set the adapter's MAC address. (Use with caution.)

-b *base*
Specify the RAM base address of the adapter. This option overrides the automatic RAM address detection, and should rarely be used.

-e *expon_backoffs*
The number of exponential backoffs before failing a transmission.

-f *tx_forget_time*
After a driver-level NACK is received from a remote node, **Net.ether7231** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which **Net.ether7231** stops throttling its output. **-f** allows you to change the default 10 seconds above.

-h *hang_ticks*
Specify the number of 50ms ticks to wait until the driver concludes that the chip is hung and resets it. Default: 10 (500ms).

-i *irq*
Use the specified interrupt line (IRQ). This option overrides automatic IRQ detection, and should rarely be used.

-l *log_net_id*
("el") Connect to the specified logical network (default is 1).

-M Enable reception of multicast packets.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times (default is 10).

-p *port* Use the given base I/O port address of the adapter, specified as a hexadecimal number *without* a leading **0x**. This option overrides the automatic I/O port detection.

-P Run promiscuously: buffer every packet, whether or not it's destined for this node. (A node usually discards any packet not addressed to it.)

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 10000000. This information is used by the Network Manager (**Net**) only when multiple drivers are running.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries (default is 20).

-v Be verbose; print out hardware info on the console when starting up.

Description:

The **Net.ether7231** driver communicates directly with the AT&T7231 chip in the onboard AT&T Lanpacer adapter used in the NCR 7450 cash register. It provides the Network Manager (**Net**) with reliable data transfer over an Ethernet network.

By default **Net.ether7231** reads the Lanpacer's configuration from the NCR 7450's CMOS RAM configuration area. This is the recommended method of operation, since it allows changing of the

settings in the CMOS configuration utility without requiring changes to the driver's command-line parameters.

The **-t** *tx_retry_ticks* option tells the driver how long to wait before considering an attempted transmission a failure; the **-n** *tx_num_retries* option tells the driver how many times it should retry transmission before giving up.

The **-e** *expon_backoff* option, alters the number of exponential backoffs the Lanpacer will execute before reporting excessive collisions and failing the transmission attempt. Changing this value from the default (16) contravenes IEEE 802.3. Therefore, specifying this option is not recommended.

Examples:

Use automatic detection of all parameters (will attach to logical LAN 1):

```
Net.ether7231 &
```

Attach to logical LAN 2:

```
Net.ether7231 -l2 &
```

Signals:

The **Net.ether7231** driver ignores SIGPWR.

Exit status:

0	Successful
>0	An error occurred

© 2005, QNX Software Systems

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.ether8003

WD/SMC 8003 Ethernet network driver (QNX)

Syntax:

Net.ether8003 [*options***] &**

Options:

-8 Force 8-bit operation on 16-bit cards to reduce memory conflicts.

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.ether8003** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which **Net.ether8003** stops throttling its output. **-f** allows you to change the default 10 seconds above.

-h *hang_ticks* Specify the number of 50ms ticks to wait until the driver concludes that the chip is hung and resets it. Default: 10 (500ms).

-i *irq* Use the specified interrupt line (IRQ). This option overrides automatic IRQ detection, and should rarely be used.

-l *log_net_id* ("el") Connect to the specified logical network (default is 1).

-m *mem* Use the specified memory segment (hexadecimal; no leading **0x**). This option overrides automatic RAM detection, and should rarely be used.

-M Enable reception of multicast packets.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times (default is 3).

-p *io_port* Use the given base I/O port address, specified as a hexadecimal number *without* a leading **0x**. This option overrides the automatic I/O port detection.

-P Run promiscuously:buffer every packet, whether or not it's destined for this node.

This option may be used in conjunction with **netsniff** to monitor other traffic on the network.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 10000000. This information is used by the Network Manager (**Net**) only when multiple drivers are running.

-s *ram_size* Use the specified length (given in Kbytes) for the shared RAM segment on the card. This option overrides auto-detect, and should rarely be needed.

-S *ps2_slot* Specify PS/2 slot when using multiple microchannel cards.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries (default is 20).

-v Be verbose; print out hardware info on the console when starting up.

Description:

The **Net.ether8003** network driver communicates directly with a WD/SMC 8003, 8013, 8416 Elite or Ultra series Ethernet network card. It provides the Network Manager with reliable data transfer over an Ethernet network.

By default, **Net.ether8003** detects all hardware parameters automatically. However, you can specify any or all of these parameters by using the options described above.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether8003** file.

If you install multiple Ethernet cards, you must specify a unique logical network ID (**-l** option) for every driver. On an ISA Bus computer, you must also specify a unique base I/O port (**-p** option) for every driver.

Examples:

Use automatic detection of all parameters for a single WD/SMC 8003, 8013, Elite or Ultra card:

```
Net.ether8003 &
```

Specify I/O port 0x220:

```
Net.ether8003 -p 220 &
```

One card is using I/O port 0x240 and logical network ID 1; the other is using I/O port 0x300 and logical network ID 2. Use automatic memory and IRQ detection for both:

```
Net.ether8003 -p 240 -l 1 &
Net.ether8003 -p 300 -l 2 &
```

Files:

Net.ether8003 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

© 2005, QNX Software Systems

Signals:

The **Net.ether8003** driver ignores SIGPWR.

Exit status:

Net.ether8003 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether8003** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.ether82557

Intel 82557/8 Ethernet network driver (QNX)

Syntax:

Net.ether82557 [-F] [-f *tx_forget_time*]
[-i *irq*] **[-l** *log_net_id*] **[-M]**
[-n *tx_num_retries*] **[-I** *pci_index*]
[-P] [-p *io_port*] **[-r** *media_rate*]
[-s *speed*] **[-t** *tx_retry_ticks*]
[-v] &

Options:

-a *io_port*
This option is no longer valid in QNX 4.25 and later. Use **-p** instead.

-f *tx_forget_time*
After a driver-level NACK is received from a remote node, **Net.ether82557** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 2 seconds, after which the eternally optimistic **Net.ether82557** stops throttling its output. The **-f** option allows you to change the default 2 seconds above.

-F
Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-I *pci_index*
Have the driver look at this specific PCI Index for an 82557 compatible card. You should only have to specify -I if you have more than one 82557 compatible card installed. (e.g. **-I0**, **-I1**, **-I2**).

-i *irq*
Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect does not work on an ISA card, you have a hardware conflict. This option should rarely be used.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 7.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address. In earlier versions of QNX, the **-p** option was used to specify the PCI index. Use the **-I** option to specify the PCI Index of the card being used.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-s *speed* Set the speed at which the card should run. Only possible with 100Mbit cards. (e.g. **-s10** or **-s100**) This disables the auto-negotiation capability of ethernet card.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries. The default is 6.

-v Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.ether82557** network driver communicates with Intel 82557/82558, and compatible PCI ethernet adapters to provide the Network Manager with reliable data transfer over an Ethernet network. The driver supports PHY auto-negotiation with auto-negotiating partners. Auto-negotiation can be disabled by using the **-s** (speed) command line option. Full-duplex operation can also be forced, using the **-F** command line option.

Most of the time, you should not need to specify the I/O port or the hardware interrupt to **Net.ether82557** — it should autodetect these parameters from the PCI BIOS.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether82557** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one 82557 compatible PCI network card, you must also specify the **-I** option for every invocation of **Net.ether82557**. (Use **-I0** for the first card, **-I1** for the second, etc.)

Examples:

Start the **Net.ether82557** driver with default parameters:

```
Net.ether82557 &
```

Start the **Net.ether82557** driver and force the speed to 100Mbit:

```
Net.ether82557 -s100  &
```

Start two copies of the driver running on separate cards; one on logical network 1, the other on logical network 2:

```
Net.ether82557 -I0 -l1 &
Net.ether82557 -I1 -l2 &
```

Files:

Net.ether82557 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether82557** driver ignores SIGPWR.

Exit status:

Net.ether82557 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether82557** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.ether82595

Intel 82595 Ethernet network driver

Syntax:

Net.ether82595 [-A] [-B] [-f *tx_forget_time***]**
[-F] [-i *irq***]**
[-l *log_net_id***] [-M]**
[-n *tx_num_retries***] [-P]**
[-p *io_port***] [-r** *media_rate***]**
[-R] [-s *speed***]**
[-t *tx_retry_ticks***] [-v] &**

Options:

-a *io_port*
This option is no longer valid in QNX 4.25 and later. Use **-p** instead.

-A
Use the AUI port on the card. Disables auto selection.

-B
Use the BNC port on the card. Disables auto selection.

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.ether82595** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 3 seconds, after which the eternally optimistic **Net.ether82595** stops throttling its output. The **-f** option allows you to change the default 3 seconds above.

-F
Force full-duplex operation of the card. The default is half-duplex.

-i *irq*
Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect does not work on an ISA card, you have a hardware conflict. This option should rarely be used.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 7.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 10000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-R The card is Removable (PCMCIA). Setting this command line option will cause the driver to monitor the PCMCIA manager for removal of the card. If the card is removed, the driver will terminate.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries. The default is 6.

-v Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.ether82595** network driver communicates with a Intel 82595 or compatible network adapter to provide the Network Manager with reliable data transfer over an Ethernet network.

Most of the time, you should not need to specify the I/O port or the hardware interrupt to **Net.ether82595** — it should autodetect these parameters.

Full-duplex operation can also be forced, using the **-F** command line option.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether82595** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one 82595 compatible PCI network card, you must specify the **-p** option on each invocation of **Net.ether82595**.

Examples:

Start the driver and let it autodetect the hardware:

```
Net.ether82595 &
```

Start the driver on an 82595 compatible PCMCIA adapter, specifying the port and IRQ:

```
Net.ether82595 -p340 -i5 -R &
```

Start two **Net.ether82595** drivers for two adapters installed in one machine. One card is using I/O port 300h and IRQ 10 on logical network 1. The second card is using I/O port 340 and IRQ 12 on logical network 2.

```
Net.ether82595 -p300            &
Net.ether82595 -p340 -i12  -l2  &
```

© 2005, QNX Software Systems

Files:

Net.ether82595 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether82595** driver ignores SIGPWR.

Exit status:

Net.ether82595 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether82595** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.ether8900

Crystal Semiconductor CS8900 Ethernet network driver (QNX) — Obsolete

Description:

Since QNX 4.25, the **Net.ether8900** driver is obsolete and has been replaced by **Net.crys8900**.

Net.ether9000

© 2005, QNX Software Systems

SMC91C90/91C92/91C100FD Ethernet network driver (QNX)

Syntax:

Net.ether9000 [*options***] &**

Options:

-c Program the NIC to monitor the carrier when transmitting.

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.ether9000** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which the eternally optimistic **Net.ether9000** stops throttling its output. The **-f** option allows you to change the default 10 seconds above.

-F Enable full-duplex operation.

-h *hang_ticks* Specify the number of 50ms ticks to wait until the driver concludes that the chip is hung and resets it. Default: 10 (500ms).

-i *irq* Specify hardware interrupt (default is to autodetect)

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast receive mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 3.

-o	Use fiber optic links.
-p *io_port*	Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the autodetect.
-P	Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode. You may use this option in conjunction with **netsniff** to monitor other traffic on the network.
-r *media_rate*	Advertise the specified bit-transmission rate. The default is 10000000. This information is used by the Network Manager when multiple network drivers are running.
-s *speed*	Specify the data rate, e.g. **-s10** or **-s100**. Default is auto-negotiate. Use of this option disables the auto-negotiation capability of the ethernet card.
-t *tx_retry_ticks*	The number of 50 millisecond intervals between transmit retries. The default is 20.
-T [**a**|**t**]	Specify connection type. If **t** is specified, the connection type is 10BASE-T. If **a** is specified, the connection type is AUI/coax.
-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.ether9000** network driver communicates directly with a SMC91C90, SMC91C92, or SMC91C100FD compatible Ethernet network card. It provides the Network Manager with reliable data transfer over an Ethernet network.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether9000** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver. If you install more than one SMC9000 or compatible network card, you must also specify the **-p** options for every invocation of **Net.ether9000** (except the one with the lowest I/O port).

Examples:

On an SMC91C92, scan for card and autodetect IRQ:

```
Net.ether9000  &
```

On an SMC91C92, use I/O port 340h and autodetect IRQ:

```
Net.ether9000 -p 340 &
```

One card is using I/O port 300h (scan will detect first) on logical network 1. The second card is using I/O port 320h on logical network 2:

```
Net.ether9000 &
Net.ether9000 -p 320 -l 2  &
```

Files:

Net.ether9000 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether9000** driver ignores SIGPWR.

Exit status:

Net.ether9000 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether9000** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.ether905

© 2005, QNX Software Systems

3Com 905 Ethernet network driver (QNX)

Syntax:

Net.ether905 **[-d** *device_id***]** **[-F]** **[-f** *tx_forget_time***]**
[-I *pci_index***]** **[-i** *irq***]**
[-l *log_net_id***]** **[-M]**
[-n *tx_num_retries***]** **[-P]**
[-p *io_port***]** **[-r** *media_rate***]**
[-R] **[-s** *speed***]**
[-t *tx_retry_ticks***]** **[-v]** **&**

Options:

-a *io_port*
: This option is no longer valid in QNX 4.25 and later. Use **-p** instead.

-d *device_id*
: Start the driver on a specific *device_id*. For example, **-d9055** would start the driver on a 905B card.

-f *tx_forget_time*
: After a driver-level NACK is received from a remote node, **Net.ether905** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which the eternally optimistic **Net.ether905** stops throttling its output. The **-f** option allows you to change the default 10 seconds above.

-F
: Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-I *pci_index*
: Have the driver look at this specific PCI Index for a 905 compatible card. You should only have to specify **-I** if you have more than one 905 compatible card installed. (e.g. **-I0**, **-I1**, **-I2**)

-i *irq*
Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect does not work on an ISA card, you have a hardware conflict. This option should rarely be used.

-l *log_net_id*
("el") Connect to the specified logical network. The default is 1.

-M
Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*
After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 7.

-P
Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port*
Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address. In earlier versions of QNX, the **-p** option was used to specify the PCI index. Use the **-I** option to specify the PCI Index of the card being used.

-r *media_rate*
Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-s *speed*	Specify the data rate, e.g. **-s10** or **-s100**. Default is auto-negotiate. Use of this option disables the auto-negotiation capability of the ethernet card.
-t *tx_retry_ticks*	The number of 50 millisecond intervals between transmit retries. The default is 6.
-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.ether905** network driver communicates with 3Com's 3C905-TX PCI Ethernet adapter to provide the Network Manager with reliable data transfer over an Ethernet network.

The driver supports PHY auto-negotiation with auto-negotiating partners. Auto-negotiation can be disabled by using the **-s** (speed) command line option.

Most of the time, you should not need to specify the I/O port

Full-duplex operation can also be forced, using the **-F** command line option.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.ether905** file.

Most of the time, you should not need to specify the I/O port or the hardware interrupt to **Net.ether905** - it should autodetect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one 905 compatible PCI network card, you must specify the **-p** option on each invocation of **Net.ether905**. If more than one 905 compatible PCI card is installed, you will need to use the **-I** option (**-I0** for the first card, **-I1** for the second, etc.)

Examples:

Start the driver and let it autodetect the hardware:

```
Net.ether905 &
```

Start the **Net.ether905** driver and force the speed to 100Mbit:

```
Net.ether905 -s100  &
```

Start two **Net.ether905** drivers for two adapters installed in one machine:

```
Net.ether905 -I0 -l1  &
Net.ether905 -I1 -l2  &
```

Files:

Net.ether905 closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ether905** driver ignores SIGPWR.

Exit status:

Net.ether905 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ether905** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

© 2005, QNX Software Systems

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Syntax:

Net.fd [-a phys_addr] [-c] [-d *dev_rate***] [-f** *tty***] [-h]**
[-i *timeout***] [-k** *comp_type***] [-l** *log_net_id***]**
[-m *min_com_length***] [-n** *tx_num_retries***]**
[-r *media_rate***] [-t** *tx_retry_ticks***] [-v]** *tty* **&**

Options:

-a *phys_addr* — Use this 48-bit hex Ethernet-style physical address. Usually, **Net.fd** gets this from **Net**'s netmap table.

-c — Don't append and check for CRC.

-d *dev_rate* — Use this device media rate for timeout calculations. By default the driver uses the baud rate of the device.

-f *tty* — Use this device for network input and output. Specify **-** for standard input.

-h — Ignore SIGHUP. By default the driver terminates on SIGHUP. This signal is sent when, for example, the carrier is lost from a modem.

-i *timeout* — Use this interbyte timeout (in tenths of seconds). The default is 10, i.e. 1 second.

-k *comp_type* — Use compression, specifying type **1** or **2**. Type **1** compresses the entire packet, including header. Type **2** compresses only the data, leaving the header uncompressed. This leaves the source and destination addresses in the header intact so that the process serving **Net.fd** can decide where to route the packet. Default is no compression.

-l *log_net_id* — ("el") Connect to this logical network (default is 1).

-m *min_com_length*

Compress packets that are this length or longer (default is 12).

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times (default is 3).

-r*media_rate* Advertise this bit-transmission rate. By default, the driver automatically calculates this value. The Network Manager (**Net**) uses this information when multiple drivers are running.

-t *tx_retry_ticks* Wait this many 50-millisecond intervals between transmit retries. By default, the driver automatically calculates this value, based on the length of the transmission and on the media rate.

-v Be verbose: print out configuration info on the console when starting up.

tty Use this device for network input and output (default is **/dev/ser1**).

Description:

The **Net.fd** network driver uses file-descriptor based I/O (i.e. *open()*, *read()*, *write()*, and so on) to receive and transmit packets. It provides the Network Manager (**Net**) with reliable data transfer over any media supported by a file-descriptor-based server process.

For example, you could use **Net.fd** to connect two machines with a null-modem RS-232 serial cable. By using file descriptor I/O to the serial devices, **Net.fd** would implicitly use **Dev**'s serial driver (**Dev.ser**) and set up a logical network link.

You should read the technote contained in the **/etc/readme/technotes/Net.fd** file.

If you start more than one network driver on the same machine, you must specify a unique network ID (**-l** option) to every driver.

Examples:

Use the first serial port (**/dev/ser1**) as logical network 1:

```
Net.fd &
```

Use the second serial port as logical network 1:

```
Net.fd /dev/ser2 &
```

Use the first serial port as logical network 2:

```
Net.fd -l 2 &
```

Support two logical networks: use the first serial port as logical network 1 and use the second serial port as logical network 2:

```
Net.fd &
Net.fd -l 2 /dev/ser2 &
```

Files:

Net.fd closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Net.fd opens and uses the *tty* named on the command line for both reading and writing.

© 2005, QNX Software Systems

Signals:

The **Net.fd** driver ignores SIGPWR, but will terminate in SIGHUP unless the **-h** option is specified.

Exit status:

Net.fd terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.fd** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Dev, **Net**, **Net.***, **netinfo**

Installation & Configuration

Net.fddidfe

DEC FDDI network driver (QNX)

Syntax:

Net.fddidfe [-f *tx_forget_time***] [-I** *pci_index***]**
[-l *log_net_id***] [-m** *burst holdoff***]**
[-n *tx_num_retries***] [-r** *media_rate***]**
[-s *slot_number***] [-t** *tx_retry_ticks***]**
[-v] [-P] &

Options:

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.fddidfe** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for 10 seconds, after which the eternally optimistic **Net.fddidfe** stops throttling its output. The **-f** option allows you to change the default 10 seconds above.

-I *pci_index* PCI index of adapter (default is 0).

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-m *burst holdoff* Set the minimum number of bus cycles between bus master requests from the adapter. The burst holdoff can be increased up to a maximum of 31, to prevent the adapter from utilizing all of the EISA bus's bandwidth. The minimum burst holdoff (and default) is 2.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 7.

-P	Program the adapter for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode. You may use this option in conjunction with **netsniff** to monitor other traffic on the network.
-r *media_rate*	Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager (**Net**) when multiple network drivers are running.
-s *slot_number*	Use the adapter in the specified EISA slot. This option overrides the default auto-detect, which scans all slots. If you have two FDDI cards installed, you must specifically tell each driver which card to talk to with the **-s** option.
-t *tx_retry_ticks*	The number of 50 millisecond intervals between transmit retries. The default is 6.
-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.fddidfe** network driver communicates directly with a DEC FDDI network controller for EISA and PCI machines. **Net.fddidfe** provides the Network Manager (**Net**) with reliable data transfer over an FDDI network.

Before installing your card, you should read the technote contained in the **/etc/readme/technotes/Net.fddidfe** file.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one FDDI network card, you must specify the **-s** option for every invocation of **Net.fddidfe**.

Examples:

Auto-detect the adapter:

```
Net.fddidfe &
```

One card is in slot 1 on logical network 1. The second card is in slot 3 on logical network 2:

```
Net.fddidfe -s 1 &
Net.fddidfe -s 3 &
```

Files:

Net.fddidfe closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.fddidfe** driver ignores SIGPWR.

Exit status:

Net.fddidfe terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.fddidfe** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.i82540

© 2005, QNX Software Systems

Intel 82540 Gigabit Ethernet network driver (QNX)

Syntax:

Net.i82540 [-f *tx_forget_time***] [-F] [-i** *irq***]**
[-I *pci_index***] [-l** *log_net_id***] [-M]**
[-n *tx_num_retries***] [-p** *io_port***] [-P]**
[-r *media_rate***] [-s** *data_rate***]**
[-t *tx_retry_ticks***] [-v] &**

Options:

-f *tx_forget_time*

The number of seconds until a received NAK is forgotten about for transmitting.

-F Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-i *irq* Program the card to use the specified hardware interrupt. This option overrides the default autodetect.

-I *pci_index* Have the driver look at this specific PCI Index for an i82540-compatible card. You should have to specify **-I** only if you have more than one i82540 card installed (e.g. **-I0**, **-I1**, **-I2**).

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast Rx mode. All packets with multicast destination addresses are buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. The default is 3.

-p *io_port*	Use the specified I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address.
-P	Program the NIC for promiscuous Rx mode. All packets are buffered, regardless of their destination addresses. The default is nonpromiscuous mode.
-r *media_rate*	Advertise the specified bit-transmission rate. The default is 100000000. The Network Manager uses this information when multiple network drivers are running.
-s *speed*	Set the speed at which the card should run. You can do this only with 100Mbit cards (e.g. **-s10** or **-s100**). This disables the auto-negotiation capability of the Ethernet card.
-t *tx_retry_ticks*	The number of 50ms ticks between Tx retries. The default is 20.
-v	Be verbose; display hardware configuration information on the screen when starting up.

Description:

The **Net.i82540** network driver has been tested with Intel's i82540 PCI ethernet adapter. The driver supports PHY auto-negotiation with auto-negotiating partners. You can disable auto-negotiation by using the **-s** (speed) command-line option. You can also use the **-F** option to force full-duplex operation.

Gigabit isn't supported — only 10 and 100 M/s.

Net.i82540 provides the Network Manager with reliable data transfer over an Ethernet network.

Most of the time, you shouldn't need to specify the I/O port or the hardware interrupt to **Net.i82540**; it should autodetect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID (using the **-l** option) to every network driver. If more than one i82540 PCI card is installed, you need to use the **-I** option (**-I0** for the first card, **-I1** for the second, and so on).

Examples:

Start the **Net.i82540** driver with default parameters:

```
Net.i82540 &
```

Start the **Net.i82540** driver and force the speed to 100M/s:

```
Net.i82540 -s100 &
```

Start two copies of the driver running on separate cards, one on logical network 1, the other on logical network 2:

```
Net.i82540 -I0 -l1 &
Net.i82540 -I1 -l2 &
```

Exit status:

0 Success.

>0 An error occurred.

See also:

Net, **Net.ether1000**, **Net.ether503**, **Net.ether8003**, **netinfo**, **netmap**

Installation & Configuration

Net.ns83815

National Semiconductor DP83815 network driver (QNX)

Syntax:

Net.ns83815 **[-F]** **[-I** *pci_index***]** **[-i** *irq***]**
[-k] **[-L** *Net_Manager_name***]** **[-l** *log_net_id***]** **[-M]**
[-m *MAC_ADDR***]** **[-n** *tx_num_retries***]**
[-P] **[-p** *io_port***]** **[-R** *num_rx***]** **[-r** *media_rate***]**
[-s *speed***]** **[-T** *num_tx***]** **[-t** *tx_retry_ticks***]** **[-v]**
[-x *vendor_ID***]** **[-y** *device_ID***]** **&**

Options:

-F Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-I *pci_index* Have the driver look at this specific PCI Index for a DP83815-compatible card. You should have to specify **-I** only if you have more than one DP83815-compatible card installed (e.g. **-I0**, **-I1**, **-I2**).

-i *irq* Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect doesn't work on an ISA card, you have a hardware conflict. This option should rarely be used.

-k Use store-forward mode. Copy complete message to NIC before transmitting.

-L *Net_Manager_name*

The default is **qnx/net**.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

© 2005, QNX Software Systems

-m *MAC_ADDR* Use this Ethernet physical address.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 3.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address. In some earlier versions of QNX, the **-p** option was used to specify the PCI index. Use the **-I** option to specify the PCI Index of the card being used.

-R *num_rx* Specify this number of rx descriptors. The default is 32.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-s *speed* Set the speed at which the card should run. Only possible with 100Mbit cards. (e.g. **-s10** or **-s100**) This disables the auto-negotiation capability of Ethernet card.

-T *num_tx* Specify this number of tx descriptors. The default is 32.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries. The default is 6.

-v	Be verbose; print out hardware configuration info on the screen when starting up.
-x *vendor_ID*	Use this vendor ID.
-y *device_ID*	Use this device ID.

Description:

The **Net.ns83815** network driver communicates with a National Semiconductor DP83815 adapter to provide the Network Manager with reliable data transfer over an Ethernet network. The driver supports PHY auto-negotiation with auto-negotiating partners. Auto-negotiation can be disabled by using the **-s** (speed) command-line option. Full-duplex operation can also be forced, using the **-F** option.

Most of the time, you shouldn't need to specify the I/O port or the hardware interrupt to **Net.ns83815** — it should autodetect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one DP83815-compatible PCI network card, you must also specify the **-I** option for every invocation of **Net.ns83815**. (Use **-I0** for the first card, **-I1** for the second, etc.)

Examples:

Start the **Net.ns83815** driver with default parameters:

```
Net.ns83815 &
```

Start the **Net.ns83815** driver and force the speed to 100Mbit:

```
Net.ns83815 -s100  &
```

Start two copies of the driver running on separate cards; one on logical network 1, the other on logical network 2:

```
Net.ns83815 -I0 -l1 &
Net.ns83815 -I1 -l2 &
```

Files:

Net.ns83815 closes its standard input, standard output, and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.ns83815** driver ignores SIGPWR.

Exit status:

Net.ns83815 terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.ns83815** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Syntax:

Net.rtl [-F] [-I *pci_index***] [-i** *irq***] [-k] [-l** *log_net_id***] [-M] [-m** *mac***] [-P] [-p** *io_port***] [-r** *media_rate***] [-s** *speed***] [-v] &**

Options:

-F Force full-duplex mode. Only full-duplex media will be used with autodetection on. When combined with the **-s** *speed* option, can force exclusive use of 10bTFD or 100bTFD, regardless of whether or not a valid link is available.

CAUTION: Full-duplex mode on half-duplex links — or vice versa — can cause networking problems.

-I *pci_index* Forces PCI detection of the driver to this *pci_index*. All PCI cards with the same PCI IDs have an index number that lets you specify the card you want. Use in association with **show_pci**. Normally, you would specify **-I** only if you have more than one Realtek card in your system.

-i *irq* This lets you override the IRQ read back from PCI space with this number. This instance of the Realtek driver will attach its interrupt handler to this interrupt.

CAUTION: Don't force **Net.rtl** to attach to an incorrect interrupt number. Interrupts enabled from the card can cause spurious interrupts to another system driver.

-k Force store/forward mode on transmission. Use this option on machines with bad PCI chipsets. It forces the complete copy of **tx** data to the **tx** FIFO before beginning transmission of the data.

© 2005, QNX Software Systems

By default, the driver uses an adaptive technique in an attempt to gain performance. If it fails, store/forward mode will automatically be chosen.

-l *log_net_id* ("el") Specify the Logical Network ID for this instance of the driver. (Default: 1)

-M Enable receipt of all multicast packets. All packets with multicast destination addresses will be buffered.

-m *mac* Specify MAC address (i.e. override the address read from the card). Use this option only if you know what you're doing!

CAUTION: Think twice before specifying a MAC address — your network may end up with multiple cards/nodes with the same physical address!

-P Enable promiscuous **rx** mode. This allows the receipt of *all* packets on this network, which will affect driver performance. All packets will be buffered, regardless of the packet's destination address.

-p *io_port* Select the Realtek board in your system with this *io_port* (I/O base address), specified as a hexadecimal number *without* a leading **0x**, to be used for this instance of the driver.

PCI dynamically assigns I/O base addresses — your card may be at a different address the next time you reboot your machine.

-r *media_rate* Allow the initial settings of *media_rate* (bit-transmission rate), which will be interpreted by **Net** at startup. The driver will override this value once a *media_rate* has been chosen, but **Net** will have already accepted this value at this point.

-s *speed* Set the speed to 10bT or 100bT, regardless of whether or not a valid link is available.

Make sure you've selected the correct *speed*. If not, you won't be able to talk to the network.

-v Enable verbose output from the driver:

- **-v** Level 1 verbose messages (PHY Link up/down messages)
- **-vv** Level 2 verbose messages (almost everything)

Description:

Net.rtl offers support for any card currently based on the Realtek Semiconductor RTL8139A chip. The Realtek RTL8139A is a highly integrated single-chip fast Ethernet controller that provides 32-bit performance, PCI bus master capability, and full compliance with IEEE 802.3u 100BASE-T specifications.

This version of the driver can be identified by using **sin ver** — it should be reported as 4.25F. Note also that the hardware is supported only in PCI I/O access mode on the PCI bus — memory-mapping isn't supported at this time.

PHY support

The **Net.rtl** driver uses the generic PHY support provided by the MDI interface in QNX. This provides specific support for the following PHYs and generic support for all other PHY chips:

Vendor	Device ID
National Semiconductor	DP83840
National Semiconductor	DP83843

continued...

Vendor	Device ID
Level One	LXT9746
Quality Semiconductor	QS6612
Integrated Circuit Systems	ICS1890
Intel	I82555
Davicom	DM9101
MYSON	MTD972

Wrong speed?

If **Net.rtl** gets the link speed wrong:

1. Disconnect a cable. This will cause a Link Down in a couple of seconds.
2. Reconnect the cable. This should return you to the correct link, forcing autonegotiation to reoccur.

Net.rtl implements a link-detection algorithm if no link is found. This sometimes can cause an incorrect link to be chosen between autonegotiation partners. Upon removal of a link, **Net.rtl** will then attempt to autonegotiate to get the correct link this time.

If a PHY is used and link speeds with duplex is forced, these settings are always used and a good link will be reported regardless of whether or not a valid link is available!

Examples:

Start **Net** and **Net.rtl** drivers for two cards in a system:

```
Net &
Net.rtl -v &
Net.rtl -I1 -l2 -vv &
```

Set interrupts:

```
Net &
Net.rtl -i10 &
```

Set speed:

```
Net &
Net.rtl -s100 -v &              # 100bT
Net.rtl -s10 -I1 -l2 -v &       # 10bT
```

Why verbose output? It can be useful as a debugging or informational aid. With only level 1 and 2 verbose modes, link state messages and some other output is produced. The highest level of verbose output is/was used as a debugging tool on releases.

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.sis9

© 2005, QNX Software Systems

SiS 900 Ethernet network driver (QNX)

Syntax:

Net.sis9 [-p *IoBase***] [-i** *IrqNum***] [-m** *MAC_ADDR***]**
[-V *Port Speed***] [-M] [-n** *TxMaxRetries***] [-P]**
[-r *MediaRate***] [-t** *TxRetryTicks***] [-s** *Speed***]**
[-F] [-l *NetId***] [-L** *NetManagerName***]**
[-I *PciIndex***] [-T** *NumTx***] [-R** *NumRx***] [-v] [-k]**
[-x *VendorId***] [-y** *DeviceId***]**

Options:

-F Force full-duplex operation of the card. The default is half-duplex, unless the PHY auto-negotiates full-duplex.

-I *PciIndex* Have the driver look at this specific PCI Index for a SiS900-compatible card. You should have to specify **-I** only if you have more than one SiS900-compatible card installed (e.g. **-I0**, **-I1**,' **-I2**).

-i *IrqNum* Program the card to use the specified hardware interrupt. This option overrides the default autodetect.

-k Store/Forward on Tx. The chip will load the complete message into its buffer before starting transmission.

-L *NetManagerName*

The name of the network manager. The default is **qnx/net**.

-l *NetId* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast Rx mode. All packets with multicast destination addresses will be buffered.

-m *MAC_ADDR* Override the default Ethernet physical address

-n *TxMaxRetries*

After failing to transmit to a remote node, retry the transmission no more than this number of times. The default is 3.

-P Program the NIC for promiscuous Rx mode. All packets will be buffered, regardless of the packet's destination address. The default is nonpromiscuous mode.

-p *IoBase* Use the specified I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detection, which uses the PCI BIOS to return a port address.

-R *NumRx* The number of Rx descriptors (default: 32).

-r *MediaRate* Advertise the specified bit-transmission rate. The default is 100000000. The Network Manager uses this information when multiple network drivers are running.

-s *Speed* Set the speed at which the card should run. You can use this option only with 100Mbit cards (e.g. **-s10** or **-s100**). This option disables the auto-negotiation capability of the Ethernet card.

-T *NumTx* The number of Tx descriptors (default: 16).

-t *TxRetryTicks* The number of 50ms ticks between Tx retries (default : 20).

-V *Port Speed* Send output to the given serial port, at the given speed (debugging). Use a space to separate the *Port* and *Speed* arguments (e.g. **-V3e8 9600**).

-v Be verbose; print out hardware configuration information on the screen when starting up.

-x *VendorId* Override the default PCI Vendor ID.

-y *DeviceId* Override the default PCI Device ID.

Description:

The **Net.sis9** network driver has been tested with SiS's 900 PCI Ethernet adapter. The driver supports PHY auto-negotiation with auto-negotiating partners. You can disable the auto-negotiation by using the **-s** command-line option. You can also force full-duplex operation by using the **-F** option.

Net.sis9 provides the Network Manager with reliable data transfer over an Ethernet network.

Most of the time, you shouldn't need to specify the I/O port or the hardware interrupt to **Net.sis9**; it should automatically detect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID to every network driver, by using the **-l** ("el") option. If more than one SiS900 compatible PCI card is installed, you'll need to use the **-I** option (**-I0** for the first card, **-I1** for the second, and so on).

Examples:

Start the **Net.sis9** driver with default parameters:

```
Net.sis9 &
```

Start the **Net.sis9** driver and force the speed to 100Mbit:

```
Net.sis9 -s100  &
```

Start two copies of the driver running on separate cards, one on logical network 1, the other on logical network 2:

```
Net.sis9 -I0 -l1 &
Net.sis9 -I1 -l2 &
```

Exit status:

0 Success.

>0 An error occurred.

See also:

Net, **Net.ether1000**, **Net.ether503**, **Net.ether8003**, **netinfo**, **netmap**

Installation & Configuration

Net.soho

© 2005, QNX Software Systems

Macronix MX98715 Ethernet network driver (QNX4)

Syntax:

Net.soho **[-F] [-f** *tx_forget_time***] [-I** *pci_index***]**
[-i *irq***] [-l** *log_net_id***] [-M]**
[-n *tx_num_retries***] [-P] [-p** *io_port***]**
[-r *media_rate***] [-s** *speed***]**
[-t *tx_retry_ticks***] [-v]**

Options:

-F Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-f *tx_forget_time*

After a driver-level **NACK** is received from a remote node, **Net.soho** temporarily throttles the transmit output on that connection because of the dropped packets. The output is reduced to approximately 400K/sec for *tx_forget_time* seconds. The default *tx_forget_time* time is 10 seconds.

-I *pci_index* Instruct the driver look at this specific PCI Index for an **MX98715**-compatible card. You would use this option only if you have more than one **MX98715**-compatible card installed (e.g. **-I0**, **-I1**, **-I2**).

-i *irq* Program the card to use the specified hardware interrupt. This option overrides the default autodetect.

-l *log_net_id* (“el”) Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast **rx** mode. All packets with multicast destination addresses will be buffered.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. The default is 7.

-P Program the NIC for promiscuous **rx** mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager when multiple network drivers are running.

-s *speed* Set the speed at which the card should run. This is possible only with 100Mbit cards (e.g. **-s10** or **-s100**). This option disables the auto-negotiation capability of your Ethernet card.

-t *tx_retry_ticks* The number of 50-millisecond intervals between **tx** retries. The default is 6.

-v Be verbose; print hardware configuration information on the screen when starting up.

Description:

The **Net.soho** driver provides the Network Manager with reliable data transfer over an Ethernet network.

The **Net.soho** network driver has been tested with Sohoware **MX98715A** PCI Ethernet adapter. The driver supports PHY auto-negotiation with auto-negotiating partners. Auto-negotiation can

be disabled by using the **-s** option. Full-duplex operation can also be forced, using the **-F** option.

Most of the time, you should not need to specify the I/O port or the hardware interrupt to **Net.soho**: the driver should autodetect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver. If more than one **MX98715**-compatible PCI card is installed, you will need to use the **-I** option: **-I0** for the first card, **-I1** for the second, etc.

Examples:

Start the **Net.soho** driver with default parameters:

```
Net.soho &
```

Start the **Net.soho** driver and force the speed to 100Mbit:

```
Net.soho -s100  &
```

Start two copies of the driver running on separate cards, one on logical network **1**, the other on logical network **2**:

```
Net.soho -I0 -l1 &
Net.soho -I1 -l2 &
```

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

Net, **Net.ether503**, **Net.ether1000**, **Net.ether8003**, **netinfo**, **netmap**

Installation & Configuration

Net.solotr

IBM PCI Token-Ring network driver (QNX)

Syntax:

Net.solotr [-f *tx_forget_time***] [-I** *pci_index***] [-i** *irq***]**
[-l *log_net_id***] [-n** *tx_num_retries***] [-r** *media_rate***]**
[-s *speed***] [-T** *table_size***] [-t** *tx_retry_ticks***] [-v] &**

Options:

-f *tx_forget_time*

After a driver-level NACK is received from a remote node, **Net.solotr** will assume that because packets were dropped, throttling the transmit output on that connection would be a good idea. So, the output will be reduced to approximately 400K/sec for two seconds, after which the eternally optimistic **Net.solotr** stops throttling its output. The **-f** allows you to change the default two-second interval.

-i *irq*

Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect doesn't work on an ISA card, you have a hardware conflict.

-l *log_net_id*

("el") Connect to the specified logical network. The default is 1.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. The default is 7.

-I *pci_index*

Have the driver look at this specific PCI Index for a **solotr**-compatible card. You should have to specify **-I** only if you have more than one **solotr**-compatible card installed (e.g. **-I0**, **-I1**, **-I2**).

-p *pci_index*	This option is no longer valid in QNX 4.25 and later. Use **-I** instead.
-r *media_rate*	Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager when multiple network drivers are running.
-s *speed*	Set the speed at which the card should run (e.g. **-s4** or **-s16**). Only possible with 100Mbit cards. This disables the Ethernet card's auto-negotiation capability.
-T *table_size*	Set the source routing table to this size. (Default: 64)
-t *tx_retry_ticks*	The number of 50-millisecond intervals between **tx** retries. The default is 6.
-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.solotr** network driver has been tested with IBM's PCI Token-Ring Adapter (P/N 41H8900). The driver supports automatic ring-speed detection. Auto-negotiation can be disabled by using the **-s** *speed* command-line option.

Net.solotr provides the Network Manager with reliable data transfer over a Token-Ring network.

Most of the time, you shouldn't need to specify the I/O port or the hardware interrupt to **Net.solotr** — it should autodetect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver. If more than one **solotr**-compatible PCI card is installed, you'll need to use the **-I** option (e.g. **-I0** for the first card, **-I1** for the second, etc.

Examples:

Start the **Net.solotr** driver with default parameters:

```
Net.solotr &
```

Start the **Net.solotr** driver and force the speed to 4Mbit:

```
Net.solotr -s4  &
```

Start two copies of the driver running on separate cards, one on logical network 1, the other on logical network 2:

```
Net.solotr -I0 -l1 &
Net.solotr -I1 -l2 &
```

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.sym

© 2005, QNX Software Systems

SymBios 53C885 network driver (QNX)

Syntax:

Net.sym [-D *port_speed***] [-F] [-I** *pci_index***] [-i** *irq***] [-k] [-L** *Net_Manager_name***] [-l** *log_net_id***] [-M] [-m** *MAC_ADDR***] [-n** *tx_num_retries***] [-P] [-p** *io_port***] [-R** *num_rx***] [-r** *media_rate***] [-s** *speed***] [-T** *num_tx***] [-t** *tx_retry_ticks***] [-v] &**

Options:

-D *port_speed* Send output to serial port (for debugging).

-F Force full-duplex operation of the card. The default is half-duplex unless the PHY auto-negotiates full-duplex.

-I *pci_index* Have the driver look at this specific PCI Index for a Sym-compatible card. You should have to specify **-I** only if you have more than one Sym-compatible card installed (e.g. **-I0**, **-I1**, **-I2**).

-i *irq* Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect doesn't work on an ISA card, you have a hardware conflict. This option should rarely be used.

-k Use store-forward mode. Copy complete message to NIC before transmitting.

-L *Net_Manager_name*

The default is **qnx/net**.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Program the NIC for multicast rx mode. All packets with multicast destination addresses will be buffered.

-m *MAC_ADDR* Use this Ethernet physical address.

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. Default is 3.

-P Program the NIC for promiscuous receive mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

You may use this option in conjunction with **netsniff** to monitor other traffic on the network.

-p *io_port* Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default auto-detect, which uses the PCI BIOS to return a port address. In some earlier versions of QNX, the **-p** option was used to specify the PCI index. Use the **-I** option to specify the PCI Index of the card being used.

-R *num_rx* Specify this number of rx descriptors. The default is 32.

-r *media_rate* Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager (**Net**) only when multiple network drivers are running.

-s *speed* Set the speed at which the card should run. Only possible with 100Mbit cards. (e.g. **-s10** or **-s100**) This disables the auto-negotiation capability of Ethernet card.

-T *num_tx* Specify this number of tx descriptors. The default is 32.

-t *tx_retry_ticks* The number of 50 millisecond intervals between transmit retries. The default is 6.

-v	Be verbose; print out hardware configuration info on the screen when starting up.

Description:

The **Net.sym** network driver communicates with a SymBios 53C885-based adapter to provide the Network Manager with reliable data transfer over an Ethernet network. The driver supports PHY auto-negotiation with auto-negotiating partners. Auto-negotiation can be disabled by using the **-s** (speed) command-line option. Full-duplex operation can also be forced, using the **-F** option.

Most of the time, you shouldn't need to specify the I/O port or the hardware interrupt to **Net.sym** — it should autodetect these parameters from the PCI BIOS.

If you install more than one network card, you must specify a unique network ID (**-l** option) to every network driver.

If you install more than one SymBios-compatible PCI network card, you must also specify the **-I** option for every invocation of **Net.sym**. (Use **-I0** for the first card, **-I1** for the second, etc.)

Examples:

Start the **Net.sym** driver with default parameters:

```
Net.sym &
```

Start the **Net.sym** driver and force the speed to 100Mbit:

```
Net.sym -s100  &
```

Start two copies of the driver running on separate cards; one on logical network 1, the other on logical network 2:

```
Net.sym -I0 -l1 &
Net.sym -I1 -l2 &
```

Files:

Net.sym closes its standard input, standard output, and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.sym** driver ignores SIGPWR.

Exit status:

Net.sym terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 **Net.sym** has shut down successfully and cleanly deregistered from **Net**.

>0 An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Net.tr164a

© 2005, QNX Software Systems

IBM Token Ring 16/4 Adapter network driver (QNX)

Syntax:

Net.tr164a [*options***] &**

Options:

-a	Assume an AT bus adapter (default is autodetect).
-d *bitmask*	Bitmask for debugging information; default: 0 (disabled).
-F *address*	Functional address the adapter should respond to (hexadecimal; no leading **0x**). See *IBM LAN Reference Manual*.
-G *address*	Group address the adapter should respond to (hexadecimal; no leading **0x**). This is a locally administered address assigned to two or more adapters to allow the adapters to copy the same frame.
-i *irq*	Specify the interrupt used. (default: autodetect)
-l *log_net_id*	("el") Connect to this logical network (default is 1).
-m	Assume a microchannel adapter (default is autodetect).
-n	Specify registered name of **Net** manager to attach to. (default: **qnx/net**)
-p *io_port*	Use this I/O port base address, specified as a hexadecimal number *without* a leading **0x**. This option overrides the driver's autodetect of 0xA20 to 0xA23 (for primary LAN cards) or 0xA24 to 0xA27 (for alternate LAN cards).
-r *media_rate*	Advertise this bit-transmission rate (default is 16000000). The Network Manager (**Net**) uses this

information when multiple network drivers are running.

-t *table_size* — Set the source routing table to this size. (Default: 64)

-v — Be verbose: print out hardware configuration info on the console when starting up.

Description:

The **Net.tr164a** network driver communicates directly with an IBM Token Ring Network 16/4 card. It works with both ISA and MCA cards that operate in shared-memory mode ("busmaster" mode isn't supported). The driver provides the Network Manager (**Net**) with reliable data transfer over a Token Ring network.

By default, **Net.tr164a** automatically locates the network adapter at either the primary (0xA20) or the alternate (0xA24) I/O port base. Once it determines the I/O port base, the driver can automatically determine all other hardware parameters.

Note that an ISA bus Token Ring card also uses I/O port 2F*n*, where *n* is the interrupt (e.g. 2, 3, 7). MCA bus cards don't use this port.

Before installing your card, you should read the technote contained in the file **/etc/readme/technotes/Net.tr164a**.

To install multiple network cards in one machine, you should specify a unique logical network ID (**-l** option) to every network driver. When installing multiple Token Ring 16/4 cards in one machine, be sure to also specify the **-p** option to every invocation of **Net.tr164a**.

Examples:

Use automatic detection of all hardware parameters:

```
Net.tr164a &
```

Do the same as above, but for a card on logical network 2:

```
Net.tr164a -l 2 &
```

© 2005, QNX Software Systems

Support two cards — one configured as the “primary LAN adapter” and connected to logical network 1, the other configured as the “alternate LAN adapter” and connected to logical network 2:

```
Net.tr164a -p A20 &
Net.tr164a -p A24 -l 2 &
```

Files:

Net.tr164a closes its standard input, standard output and standard error immediately after initialization.

Error messages are displayed via the *qnx_display_*()* functions, *not* through standard error.

Signals:

The **Net.tr164a** driver ignores SIGPWR.

Exit status:

Net.tr164a terminates only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0	**Net.tr164a** has shut down successfully and cleanly deregistered from **Net**.
>0	An error occurred during initialization.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Syntax:

Net.tulip **[-A] [-B] [-H** *media_code***]**
[-c *chipset***] [-d] [-F] [-I** *pci_index***]**
[-i *irq***] [-l** *log_net_id***] [-M]**
[-m *mac***] [-n** *tx_num_retries***] [-o]**
[-P] [-p *io_port***] [-r** *media_rate***]**
[-s] [-t *tx_retry_ticks***] [-v] [-w** *wirespeed***] [-Z] &**

Options:

-A Use AUI. Forces use of the AUI media block if it's present in the SROM. If it's not, the driver will default to media autodetection.

-p io_port Use the given I/O port base, specified as a hexadecimal number *without* a leading **0x**. This option overrides the default (**0x300**).

-B Use BNC. Forces use of the BNC media block if it's present in the SROM. If it's not, the driver will default to media autodetection.

-H *media_code* Specify media code as follows:

0	10bT
1	BNC
2	AUI
3	100bT
4	10bTFD
5	100bTFD
6	100bT4
7	100bFX
8	100bFXFD
9	MII

These settings will load the corresponding media block from the SROM, provided the block is present in the SROM. The card will operate only in the media specified. If the block isn't present, the driver will attempt autodetection of the media. Note that you can use the **tdump** program to check if the media block is present.

-c *chipset* Specify the card's chipset: 21040, 21041, 21140, 21142, 21143.

-d Disable RX errata code.

-F Enable full-duplex operation.

-o Use fiber media. Forces use of the fiber media block if it's present in the SROM. If it's not, the driver will default to media autodetection.

-i *irq* Program the card to use the specified hardware interrupt. This option overrides the default autodetect. If the autodetect doesn't work on an ISA card, you have a hardware conflict.

-s Use store/forward mode on transmission. This should be used on machines with bad PCI chipsets. It forces the complete copy of **tx** data to the **tx** FIFO before beginning transmission of the data. By default, the driver uses an adaptive technique in an attempt to gain performance. If it fails, store/forward mode will automatically be chosen.

-l *log_net_id* ("el") Connect to the specified logical network. The default is 1.

-M Enable multicast mode. This enables receipt of all multicast packets. All packets with multicast destination addresses will be buffered.

-m *mac* Specify MAC address. Use this option only if you know what you're doing!

-n *tx_num_retries*

After failing to transmit to a remote node, retry transmission no more than this number of times. The default is 7.

-P

Program the NIC for promiscuous **rx** mode. All packets will be buffered, regardless of the packet's destination address. The default is non-promiscuous mode.

-I *pci_index*

Specify PCI index. All PCI cards with the same PCI IDs have a PCI index number. This allows you to specify the card you want. Use in association with **show_pci**. You would use this option only if you have more than one tulip card in your system.

-r *media_rate*

Advertise the specified bit-transmission rate. The default is 100000000. This information is used by the Network Manager when multiple network drivers are running.

-w *wirespeed*

Force speed to be either 10bT or 100bT, regardless of link availability.

-t *tx_retry_ticks*

The number of 50-millisecond intervals between **tx** retries. The default is 6.

-v

Be verbose; print out hardware configuration info on the screen when starting up. You have these levels of verbosity:

- **-v** Level 1: PHY link up/down messages.
- **-vv** Level 2: almost everything.
- **-vvv** Level 3: allow debug output via **netinfo -Lx**.

The highest level of verbose output is used mainly as a debugging tool. For example, if you do a **netinfo -L1 > 1**, the

regular output of **netinfo** will be directed to the file **1**, and extra debugging information from the driver will appear on your screen.

-Z Zap CSR values to default. This is for all those 21143 cards with "questionable" SROM values for the CSR registers. This forces the driver to override the SROM CSR values and use the defaults. Allows 10bT to work on some of the 21143 boards.

The **-Z** option will override the SROM and insert default values — this could either enable/disable your media.

Description:

The **Net.tulip** driver should be treated as a completely new version of the **Net.Ether21x4x** driver and as such has a few known limitations (see release notes).

Net.tulip supports all cards currently supported via **Net.Ether21x4x**, including the following DEC chipsets: 21040, 21041, 21140, 21142/21143 and their various steppings.

SROM support has been added for all standard SROM formats from the early 21041 compact format through to the latest 4.05 format from Digital.

Note that this version of the driver can be identified via **sin** versions and should be reported as 4.24F or later.

PHY support

The **Net.tulip** driver uses the generic PHY support provided by the MDI interface in QNX. This includes specific support for the following PHYs (as well as generic support for all other PHY chips):

- National Semiconductor DP83840

- National Semiconductor DP83843
- Level One LXT9746
- Quality Semiconductor QS6612
- Integrated Circuit Systems ICS1890
- Intel I82555

If **Net.tulip** gets the link speed wrong, disconnect a cable to cause a *link down* in a couple of seconds. Then reconnect the cable — this should return you to the correct link (and force autonegotiation to occur again).

Net.tulip implements a link detection algorithm if no link is found. This can sometimes cause an incorrect link to be chosen between autonegotiation partners. Upon removal of a link, **Net.tulip** will then attempt to autonegotiate, getting the correct link this time round.

CAUTION: If a PHY is used and you specify link speed/duplex settings at the command line, the hardware will be configured to those settings — the PHY will report a valid link even if there isn't one! In other words, when you force link speed/duplex settings, you disable the PHY's autodetection/negotiation features.

Examples:

Here are the main options you'd use for this driver:

-c *chipset*
-H *media_code*
-I *pci_index*
-i *irq*
-l *log_net_id*
-w *wirespeed*
-vvv

Start two cards in a system, specifying a PCI index (**1**) and logical network ID (**2**) for the second card:

```
Net &
./Net.tulip -v &
./Net.tulip -I1 -l2 -vv &
```

Start a 21140-based card using IRQ 10:

```
Net &
./Net.tulip -c 21140 -i10 &
```

Set different speeds for two cards:

```
Net &
./Net.tulip -w100 -v &          # 100bT
./Net.tulip -w10 -I1 -l2 -v &   # 10bT
```

Set media codes for two cards:

```
Net &
./Net.tulip -H0 -vv &           # 10bT
./Net.tulip -H6 -I1 -l2 -v &    # 100bT4
```

Exit status:

0 Successful completion.

>0 An error occurred.

Caveats:

If the **Net.tulip** driver prints the following message on startup:

```
Rx DMA Workaround in Effect
```

then your network card may suffer from a known receive-process-hang problem. In 99.9% of the cases, normal operation will remain totally unaffected and everything will be okay. However, on some machines with older PCI chipsets, **Net.tulip** performance can be severely degraded.

You can tell if your machine is affected by running the card under network load and monitoring the output from **netinfo** at the end. If you see the following message, you know you're affected:

```
tulip ( rx) Rx Overflow Errata, discarding pkts
```

The cause of the problem is PCI DMA overruns, which result in corrupt **Rx** data packets (which then must be discarded, causing Protocol timeouts).

Most of the chips in the DEC family 21140 through to 21143 are subject to this problem (newer chips to a much lesser degree). Contact Digital for more information.

See also:

Net, **Net.***, **netinfo**

Installation & Configuration

Net.via

© 2005, QNX Software Systems

VIA Technologies VT86C100A Ethernet network driver (QNX)

Syntax:

Net.via [-F] [-I *pci_index***] [-i** *irq***] [-k] [-l** *log_net_id***] [-M] [-m** *mac***] [-P] [-p** *io_port***] [-r** *media_rate***] [-s** *speed***] [-v] &**

Options:

-F Force full-duplex mode. Only full-duplex media will be used with autodetection on. When combined with the **-s** *speed* option, can force exclusive use of 10bTFD or 100bTFD, regardless of whether or not a valid link is available.

CAUTION: Full-duplex mode on half-duplex links — or vice versa — can cause networking problems.

-I *pci_index* Forces PCI detection of the driver to this *pci_index*. All PCI cards with the same PCI IDs have an index number that lets you specify the card you want. Use in association with **show_pci**. Normally, you would specify **-I** only if you have more than one VIA card in your system.

-i *irq* This lets you override the IRQ read back from PCI space with this number. This instance of the VIA driver will attach its interrupt handler to this interrupt.

CAUTION: Don't force **Net.via** to attach to an incorrect interrupt number. Interrupts enabled from the card can cause spurious interrupts to another system driver.

-k Force store/forward mode on transmission. Use this option on machines with bad PCI chipsets. It forces the complete copy of **tx** data to the **tx** FIFO before beginning transmission of the data. By default, the driver uses an adaptive technique in

an attempt to gain performance. If it fails, store/forward mode will automatically be chosen.

-l *log_net_id* ("el") Specify the Logical Network ID for this instance of the driver. (Default: 1)

-M Enable receipt of all multicast packets. All packets with multicast destination addresses will be buffered.

-m *mac* Specify MAC address (i.e. override the address read from the card). Use this option only if you know what you're doing!

CAUTION: Think twice before specifying a MAC address — your network may end up with multiple cards/nodes with the same physical address!

-P Enable promiscuous **rx** mode. This allows the receipt of *all* packets on this network, which will affect driver performance. All packets will be buffered, regardless of the packet's destination address.

-p *io_port* Select the VIA board in your system with this *io_port* (I/O base address), specified as a hexadecimal number *without* a leading **0x**, to be used for this instance of the driver.

PCI dynamically assigns I/O base addresses — your card may be at a different address the next time you reboot your machine.

-r *media_rate* Allow the initial settings of *media_rate* (bit-transmission rate), which will be interpreted by **Net** at startup. The driver will override this value once a *media_rate* has been chosen, but **Net** will have already accepted this value at this point.

-s *speed* Set the speed to 10bT or 100bT, regardless of whether or not a valid link is available.

Make sure you've selected the correct *speed*. If not, you won't be able to talk to the network.

-v Enable verbose output from the driver:

- **-v** Level 1 verbose messages (PHY Link up/down messages)
- **-vv** Level 2 verbose messages (almost everything)

Description:

Net.via offers support for any card currently based on the VIA Technologies VT86C100 chipsets (codenamed: Rhine/Rhine II).

This version of the driver can be identified by using **sin ver** — it should be reported as 4.25F. Note also that the hardware is supported only in PCI I/O access mode on the PCI bus — memory-mapping isn't supported at this time.

PHY support

The **Net.via** driver uses the generic PHY support provided by the MDI interface in QNX. This provides specific support for the following PHYs and generic support for all other PHY chips:

Vendor	Device ID
National Semiconductor	DP83840
National Semiconductor	DP83843
Level One	LXT9746
Quality Semiconductor	QS6612

continued...

Vendor	Device ID
Integrated Circuit Systems	ICS1890
Intel	I82555
Davicom	DM9101
MYSON	MTD972

Wrong speed?

If **Net.via** gets the link speed wrong:

1. Disconnect a cable. This will cause a Link Down in a couple of seconds.
2. Reconnect the cable. This should return you to the correct link, forcing autonegotiation to reoccur.

Net.via implements a link-detection algorithm if no link is found. This sometimes can cause an incorrect link to be chosen between autonegotiation partners. Upon removal of a link, **Net.via** will then attempt to autonegotiate to get the correct link this time.

If a PHY is used and link speeds with duplex is forced, these settings are always used and a good link will be reported regardless of whether or not a valid link is available!

Examples:

Start **Net** and **Net.via** drivers for two cards in a system:

```
Net &
Net.via -v &
Net.via -I1 -l2 -vv &
```

Set interrupts:

```
Net &
Net.via -i10 &
```

© 2005, QNX Software Systems

Set speed:

```
Net &
Net.via -s100 -v &                # 100bT
Net.via -s10 -I1 -l2 -v &         # 10bT
```

Why verbose output? It can be useful as a debugging or informational aid. With only level 1 and 2 verbose modes, link state messages and some other output is produced. The highest level of verbose output is/was used as a debugging tool on releases.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

Net, **Net.***, **netinfo**, **netmap**

Installation & Configuration

Syntax:

netboot [-Aav] [-B *burst***] [-b** *address***] [-f** *bootfile***]**
[-m *mapfile***] [-P** *packets***] [-S** *ms***] &**

Options:

-a Automatically assign a logical node ID to any node that's not currently defined in the **netmap** file. A boot request is processed and the node's entry is added to the file. The assignment of logical node IDs is incremental from the highest existing logical node ID in the **netmap** file.

-A Similar to **-a**, except that each new node will be assigned the lowest unassigned logical node ID, thus filling any "holes" in the network node map.

-b *address* For any node booting an operating system built at boot time by **buildqnx** (**f=***buildfile* in that node's entry in the bootfile), pass this as the load address to **buildqnx**. The address, which is always specified in hexadecimal, must be greater than or equal to 0x1000 and must be a multiple of 0x1000.

-B *burst* The starting packet burst size (default: 40).

Boot data is sent out in bursts of packets, in between which **netboot** pauses. This is done so that machines executing from the boot ROM (which is slower than executing out of RAM) are not overwhelmed and unable to keep up with the flow of data. If a given node repeatedly tries to boot (within a short interval of the last data packet being sent), **netboot** will assume that the machine had trouble with the data rate and will use smaller packet bursts. The **netboot** utility will remember the last packet burst size used for a given node and will reuse it if that machine boots again within 15 minutes.

Special **-B** *burst* values may be specified to restrict this auto-slowdown feature:

-B 0	Disables the auto-slowdown completely. The packet burst rate is fixed to the default or whatever was specified in another **-B** option as the starting packet burst size.
-B *-minburst*	Specifying a negative number to **-B** puts a lower limit on the burst size that **netboot** will use to boot a node. The default is 2 (**-B -2**).

-f *bootfile* Use this file as the network boot configuration file. The default is **/etc/config/netboot**. The *bootfile* contains information about how to boot each node in the network that will be serviced by **netboot**.

-m *mapfile* Use this file as the network node mapfile. The default is **/etc/config/netmap**. In most cases, you should ensure that the file you specify here is the same one specified for **netmap -f** or **netmap -F**.

-P *packets* Try to read an OS image this amount at a time. The amount will be the packet size of the boot media multiplied by the specified *packets*. A large number *may* result in too large a delay between packets for some boot ROM/network card combinations. This number must be between 1 and 10. Default: 5.

-S *ms* The number of milliseconds to pause between transmission of each set of *burst* packets (see **-B**). The number will be rounded to an even multiple of the current timer resolution at the time the boot request is serviced (see **ticksize**). (1**<**=*ms***<**=1000; default: 10)

-v Be verbose; print each boot request to standard output. Multiple **-v** options will cause **netboot** to be increasingly verbose for troubleshooting.

Description:

The **netboot** utility services boot requests from machines attempting to boot over the network. If a node is being used as a boot server, **netboot** must be running on that node before other machines can boot over the network from that node.

When **netboot** starts, it queries the Network Manager (**Net**) about the network drivers currently running on the machine. Using the information it receives, **netboot** will "adopt" a corresponding set of boot packet protocols. **Net** then passes incoming boot packets to **netboot**, which services the boot request.

When boot request packets are passed to **netboot**, **Net** informs **netboot** of the following information:

- the physical node ID of the requesting node
- the logical network ID that the request came from
- the server ID (A, B, C, etc.) that the node is requesting boot service from (this ID, which is contained in the boot request packet itself, is relevant only to broadcast boot requests)

With this information, **netboot** first determines the logical node ID of the requesting node, and then looks up how it is to boot that logical node.

Determining the logical node ID

The **netboot** utility determines the logical node ID of the requesting node by looking for an entry in the **/etc/config/netmap** file that matches the logical network ID and physical node ID of the requesting node. Only the logical-to-physical node ID mapping fields are considered; that is, **netboot** will try to boot the node regardless of whether the node is masked or unmasked. For more information on

the format of the **netmap** file, see the documentation for the **netmap** utility.

If it doesn't locate a matching entry in the **/etc/config/netmap** file and if neither **-a** nor **-A** was specified, **netboot** will ignore this boot request and go back to waiting for another one. However, if **-a** or **-A** was specified, **netboot** will assign a logical node ID to the requesting node and append an entry for it to the end of the **netmap** file. It will then proceed to the next step of looking up how to boot that node.

Servicing the boot request

Once **netboot** has found out the logical node ID of the requesting node, it uses that ID in combination with the logical network ID that the request came in on to look up in the **/etc/config/netboot** bootfile for information on how the boot request should be serviced.

The **netboot** file contains lines of the form:

log_node_ID **[***log_net_ID***] [c=***build_command***] f=***file***|F=***file* **[***server_ID***]**

Each field is separated by white space consisting of space or tab characters. At least one white space character must separate each field. Any line starting with a **#** character is treated as a comment and ignored.

Don't put any comments on the same line as an entry. For example, don't do this:

```
8 f=build/node.8  #Node 8 has a UPS
```

Comments must reside on their own line, like this:

```
#Node 8 has a UPS
8 f=build/node.8
```

The fields are as follows:

log_node_ID The logical node ID, or *****, meaning "any node." Entries containing ***** should generally be placed after entries that contain more specific information, since **netboot** will stop scanning once it obtains a valid match.

log_net_ID The network the boot request came from, or *****, meaning "any network." If this field is absent, "any network" is assumed.

c=*build_command*

The command to be used to build the OS image. Default: **buildqnx**

f|F=*file* The image file or build file to be used to generate the OS required to service the boot request. This field is used by **netboot** only if the *log_node_ID*, *log_net_ID*, and *server_ID* fields have matched the incoming boot request.

This field has two forms:

F=*imagefile*

This form specifies that the named OS image be transmitted to the booting node. The *imagefile* must name the path to an OS image that has been built previously (usually via the **buildqnx** utility). If a relative pathname (one not starting with **/**) is used, it's assumed to be relative to the **/boot** directory. For example, **F=images/ws.arcnet** is equivalent to **F=/boot/images/ws.arcnet**.

f=*buildfile* **[b=***address***]**

This form instructs **netboot** to invoke **buildqnx** (or the program specified by the **c=***build_command* field) on the fly to build an OS image. This image is

then piped back into **netboot** and shipped as an image over the network. The specified *buildfile* is used by **buildqnx** to construct the image. As with the above **F=** form, relative pathnames for the buildfile are assumed to be relative to the **/boot** directory. (Note that **f=***buildfile* is the preferred form.)

You can include a [**b=***address*] after the **f=***buildfile* to specify a load address for the OS image being constructed. If a **b=** is present, it will override (for this node only) the default address or the one specified by the **-b** command-line option to **netboot**. As with the **-b** command-line option, the address is always specified in hex, must be greater than or equal to 0x1000, and must be a multiple of 0x1000.

Generally, you should use **f=***buildfile* instead of **F=***imagefile*, since you save considerable disk space by not keeping prebuilt OS images on disk. Nevertheless, you may sometimes wish to freeze a particular image to keep it immune from subsequent OS module updates. For example, if you build images on the fly (**f=**) and obtain an updated version of **Proc**, the new **Proc** will become part of that network-booted OS image the next time the node is booted. However, if you built the image beforehand and used **F=***imagefile*, the new **Proc** would not form part of the network-booted OS image until you run **buildqnx** to create a new imagefile on disk.

Since QNX is a modular OS, you don't need to build into the transmitted image all the modules you may require. Once the operating system loads, you can start many subsequent modules from the command line, or more typically, from the **sysinit.***node* file. For more information on this method, see the **buildqnx** and **sinit** utilities.

server_ID This field, which is optional, pertains only to broadcast boot requests, i.e. the style of boot request in which the booting node broadcasts a request to *all* nodes on a network. The field is ignored if the boot request isn't a broadcast type. (QNX's Ethernet boot ROM uses the broadcast method; the Corman Arcnet boot ROM doesn't.)

Broadcast boot requests are similar to saying, "Will my boot server A boot me, please?" Every **netboot** process running on the network will get the broadcast boot request. It's up to each boot server to look in its own **/etc/config/netboot** bootfile to determine what its relationship to the booting node is. Based on this relationship, the boot request will be either serviced or ignored. That is, each boot server must check its bootfile to discover what server it is to be for the requesting node (i.e. server A, B, or C) and whether this is the server that the actual boot request is asking for. If it's a match, the request is serviced. Otherwise, the request is ignored.

If **netboot** is being run on multiple nodes, each **netboot** must have its own bootfile tailored to uniquely identify it as each node's primary, secondary, or tertiary boot server. This is required because of the way servers respond to the broadcast boot requests. If it isn't done, the multiple netboot servers on the network will respond simultaneously to the broadcast boot request.

The *server_ID* field must be a single alphabetic character. Typically, network boot ROMs that broadcast boot requests ask only for A, B, or C (primary, secondary, tertiary). The field isn't case-sensitive: upper- and lowercase both mean the same thing.

For the purposes of serving broadcast boot requests, having no *server_ID* field is treated the same as

having a *server_ID* field specifying A (i.e. **netboot** will regard itself as being that node's primary boot server, by default).

Example netboot file

Let's say you have two networks connected to node 1: logical network 1 (Arcnet) and logical network 2 (Ethernet).

The following are the contents of a **/etc/config/netboot** file that could be used by **netboot** on node 1:

NID	LAN	Bootfile	"I am server..."
2		f=build/ws.arcnet	
3	1	f=build/ws.arcnet	
3	2	f=build/ws.ether	B
*	1	f=build/ws.arcnet	
*	2	f=build/ws.ether	C

The entry for logical node 2 is simple: boot node 2 regardless of which logical network the request came in on; use the build file **ws.arcnet**.

For node 3, a boot request may arrive via logical network 1 or via logical network 2. If the boot request for node 3 comes via logical network 1, then node 3 will be booted with the build file **ws.arcnet**. If the boot request for node 3 comes via network 2, **netboot** will respond to the request only if node 3 asks for its secondary (**B**) boot server. (We're assuming that another node is the primary boot server for nodes booting by Ethernet.) The **ws.ether** buildfile would be used in this case.

Any other nodes booting via network 1 will boot using the buildfile **ws.arcnet**. Any nodes that request to be booted from their tertiary (**C**) boot server on logical network 2 will boot using the build file **ws.ether**.

Examples:

Run in verbose mode on the unused console **/dev/con8**:

```
ontty /dev/con8 netboot -v &
```

Assign node IDs automatically:

```
netboot -A &
```

Caveats:

As new network drivers become available for QNX, new versions of the **netboot** utility will be released that know about the new drivers and their boot packet protocols. When **netboot** starts, it will write a warning message to standard error if it doesn't support one of the network drivers you're running. A list of supported network types (e.g. Arcnet, Ethernet) is maintained in the usage message (**use netboot**).

See also:

buildqnx, **Net**, **netmap**

netinfo

© 2005, QNX Software Systems

Display network events (QNX)

Syntax:

netinfo [-f *time_format***] [-n** *node***]**

netinfo -L *logical_net_ID* **| -l**

netinfo -a

Options:

-a Write all error code numbers and corresponding text to standard output.

-f *time_format*

Format the time column according to the specified format string. The formats allowed are the same as the display formats supported by the **date** utility (see **date**).

-l ("el") Display driver-specific information for all logical networks.

-L *logical_net_ID*

Display driver-specific information for the specified logical network.

-n *node* Display the **netinfo** log of the specified node.

Description:

The **netinfo** utility asks the Network Manager (**Net**) for the contents of its circular buffer of significant network runtime events. These events are unusual or interesting occurrences logged by either the Network Manager or the network drivers. For each of these network events, **netinfo** displays a line of output that consists of:

- the time of the network event (if you'd also like the date to be displayed, use the **-f** option)
- the node number associated with the event

- the event's numerical code
- English text that explains what the error signifies

If no node is specified via the **-n** option, **netinfo** will display network events for this node.

For a description of the strings you can use to format the date and time, see the **date** utility.

Exit status:

0 Successful.

>0 An error occurred.

See also:

date, **sin**, **tracectrl**, **traceinfo**, **tracelogger**

netmap

© 2005, QNX Software Systems

Load the Network Manager's logical-to-physical node ID mapping table (QNX)

Syntax:

netmap [-b] [-F *mapfile* **| -f] [-m** *map_entry***] [-n** *node***]**

netmap [-b] [-n *node***]**

Options:

-b	Display brouting nodes (nodes through which you have access to other LANs with which you are not directly connected).
-f	Set the network node mapping table from the default mapfile **/etc/config/netmap**. (**root** only)
-F *mapfile*	Set the network node mapping table from the specified mapfile. (**root** only.) If you specify **-F -**, the network node mapping table will be read from standard input.
-m *map_entry*	Set a nodemap entry in the network node mapping table to the specified entry. (**root** only)
-n *node*	Read/set the network node mapping table of the specified node (default is the node **netmap** is executed on).

Description:

The **netmap** utility is used to specify node mappings to the network manager (**Net**), as well as display, change, or delete mappings that **Net** is already aware of.

When the Network Manager (**Net**) starts up, it knows of no mappings between logical and physical nodes other than those specified by its **-m** option. In order for **Net** to be informed of the node mappings it will need, the **netmap** utility is normally executed in the **sysinit** file. This is done whether you're booting from a local hard disk or from across the network.

You can specify node mapping information to **netmap** via files (**-f** or **-F** options) or via the command line (**-m** option). When run without **-F**, **-f**, or **-m**, **netmap** displays the current network mapping information of the node it is run on (or of the specified node, if **-n** is used). The output will be in the same format used for the network mapping files, hence you can redirect the output to a **netmap** file or pipe it directly into a **netmap -F -**.

When **-m** is specified along with **-F** or **-f**, the entry given to **-m** will behave as if it preceded the first entry in the mapfile specified to **-F**. This mapfile is the same file **netboot** uses to map a physical node ID to a logical node ID.

Specifying a map_entry

There are two forms of *map_entry*:

- **one-to-one**
- [**delete|mask|unmask**] *mapping_info*

The first form, **one-to-one**, is a special case that treats every node as having a logical node ID that's the same as the node's physical node ID. This form is useful only for a single Arcnet network where the logical node IDs are the same as the physical node IDs. In most cases, this form of *map_entry* is placed as a single line in the mapping file rather than specified at the command line.

When specifying a *map_entry* at the command line, you must enclose the entry in quotes.

The *mapping_info* used in the second form is as follows:

logical_node_ID *logical_net_ID* *physical_node_ID*

This tells the Network Manager how the specified logical node may be reached via the specified logical network as a particular physical node ID. The logical node ID will be rejected by **Net** if it does not lie within the valid range according to the number of nodes the system is licensed for (the range is 1...*num_licenses*). The command **sin**

info will show the number of nodes the system is licensed for. Likewise, the logical network ID must be one for which a driver is running (the command **netmap -l** will display the drivers currently running). Network configuration is discussed extensively in the QNX *Installation & Configuration* manual.

The *mapping_info* can be preceded by an optional keyword to mask, unmask, or even delete a logical node completely, or to affect access to that logical node via a particular logical network driver.

When a keyword is present, the *mapping_info* is always interpreted before the keyword. For example, the following command would create a mapping for logical node 1 via logical network 2 and with 5 as the physical node ID. Having done this, **netmap** would then mask off any access to logical node 1 via logical network 2:

```
netmap -m "mask 1 2 t5"
```

Similarly, the following would allow access to logical node 1 via logical network 2 using a physical node ID of 5. The **unmask** keyword is technically unnecessary here, since setting a mapping always unmasks any previous masking of the logical node for the specified logical network:

```
netmap -m "unmask 1 2 t5"
```

Then the **delete**, **mask** or **unmask** keyword is specified, the *logical_net_ID* and *physical_node_ID* fields may be omitted. When the *logical_net_ID* is omitted, the keyword will apply to all entries for that *logical_node_ID*.

Deleting vs masking

Deleting a mapping allows the *map_entry* to be reused, while masking an entry ties up the entry, preventing its reuse.

If a packet is received and the transmitting node isn't in the Network Manager's in-memory mapping table, a mapping will automatically be added to the in-memory table only. You can prevent this automatic addition of a node's mapping by creating a **mask**ed *map_entry* for that particular node. (This automatic mapping isn't related to the **-a** and **-A** options to **netboot**, which add to the **/etc/config/netmap** file.)

Also note that **netmap** may fail to add a *map_entry* if the mapping table is full (**-n** option to **Net**) or if the total number of network drivers has been exceeded (**-d** option to **Net**).

More examples

Consider the following configuration where node 1 is connected to three networks (network 1 is Arcnet, networks 2 and 3 are Ethernet), and node 2 is connected to two of those networks (network 1 and network 2):

Since we have mixed networks, the **one-to-one** map entry can't be used. The following is a mapping file you could use for the above network:

```
1 1 t1
1 2 0000c0 0d9e40
1 3 0000c0 7b7113
2 1 t2
2 2 0000c0 4a9330
```

These mappings specify three logical networks that may be used to communicate with node 1, and two logical networks that may used to communicate with node 2.

You could do any of the following from node 2:

- Disable access to logical node 1 via logical network 2:

  ```
  netmap -m "mask 1 2"
  ```

- Mask off *all* access to logical node 1:

```
netmap -m "mask 1"
```

- Allow access to logical node 1 via all logical networks:
  ```
  netmap -m "unmask 1"
  ```

- Allow access to logical node 1 via logical network 2:
  ```
  netmap -m "unmask 1 2"
  ```

- Delete mapping information for logical node 1 via logical network 2:
  ```
  netmap -m "delete 1 2"
  ```

- Delete all mapping information for logical node 1:
  ```
  netmap -m "delete 1"
  ```

Examples:

Display mapping information on this node:

```
netmap
```

Display mapping information for logical node 2:

```
netmap -n 2
```

Load the local Network Manager with the node ID mappings contained in the file **/etc/config/netmap**:

```
netmap -f
```

Load the Network Manager on logical node 2 with the node ID mappings contained in the file **/etc/testmap**:

```
netmap -n 2 -F /etc/testmap
```

Map all logical node IDs to physical node IDs on a single Arcnet network:

```
netmap -m "one-to-one"
```

Load node mappings to match those on logical server node 1:

```
netmap -n1 | netmap -F -
```

Exit status:

0	Successful
>0	An error occurred.

See also:

buildqnx, **Net**, **netboot**

System Architecture

Installation & Configuration

netpoll

© 2005, QNX Software Systems

Display or change default polling parameters of a node (QNX)

Syntax:

netpoll [-i *idle_time***] [-n** *node***] [-p** *poll_period***]**
[-r *num_retries***]**

Options:

-i *idle_time*	The length of time, in seconds, that the system will allow a *virtual circuit* (VC) to remain idle before checking the VC's integrity (default is to not change the idle time from its current setting). The integrity check consists of transmitting a network poll packet to the remote node.
-n *node*	The node on which to change the poll parameters. The default is the node from which the utility is invoked.
-p *poll_period*	The length of time, in seconds, between checks for idle VCs once the initial *idle_time* (see **-i**) has elapsed (default is to not change the poll period).
-r *num_retries*	The number of times that the system will transmit a network poll packet to a remote node before it concludes that the remote node is permanently unreachable (default is to not change the number of retries).

Description:

In QNX, virtual circuits (VCs) provide transparent interprocess communication between processes residing on two different nodes in a network. If a node abruptly powers down, or if it can no longer be communicated with, the resources associated with the VCs to that node should be recovered.

Each node polls its own VCs, checking their integrity. The **netpoll** utility lets you view or change the default polling parameters of a specific node.

To display the polling parameters for a node, don't specify any parameters; if the node is remote, specify only the **-n** *node* option.

The initial polling parameters are determined by the version of **Net** that is being run. Typical initial values are:

Parameter	Value
poll period	10 seconds
idle time	10 seconds
num retries	6

Examples:

The following example:

- reduces to an average of 10 seconds the time allowed to detect a node "disappearing" (*poll_period* × *num_retries*), at the expense of increased CPU consumption
- increases the frequency of the VC polling
- decreases the maximum number of allowed polls:

```
netpoll  -p 2  -i 2  -r 5
```

The following example:

- increases to an average of 200 seconds the time allowed to detect a node "disappearing" (*poll_period* × *num_retries*)
- decreases the frequency of the VC polling
- increases the maximum number of allowed polls:

© 2005, QNX Software Systems

```
netpoll  -p 20  -i 20  -r 10
```

See also:

Net, **netboot**, **netinfo**

netsniff

Ethernet sniffing utility (QNX)

Syntax:

netsniff [*options***]...**

Options:

-a *src_dst_nid*	Buffer any packets that are addressed to or from the specified physical node ID (*src_dst_nid*)
-d *dst_nid*	Buffer only packets that are addressed to the specified physical destination node ID (*dst_nid*)
-D *protocol*	Don't display packets with the given protocol type. The *protocol* argument may be specified in decimal, hex (prefixed by **0x**) or octal (prefixed by **0**). Multiple **-D** options may be specified.
-f	Only fill buffer. Instead of the default continuous received packet display (with possible overflows of packet buffer) quit after packet buffer fills.
-i	Instead of the default hex/ASCII packet content display, try to interpret the packet contents.
-I	Interpret packet contents as per **-i**, but give more detail.
-L *len*	Statistics mode; *len* is the minimum length to display.
-l *log_lan*	("el") Buffer only packets received from the specified logical LAN. Default is all LANs.
-m *num_Kbytes*	Allocate the size (in kilobytes) of packet buffer RAM. Default 1024, max 32767, min 64.
-n *num_bytes*	Display the number of bytes of each packet (default 48).

-p *protocol*	Display packets of this protocol type. The *protocol* argument may be specified in decimal, hex (prefixed by **0x**) or octal (prefixed by **0**). Multiple **-p** options may be specified.
-Q	Display QNX packets. Equivalent to **-p 0x8203**.
-S	Turn on statistics mode (based on *src_nid*).
-s *src_nid*	Buffer only packets that are addressed from the specified physical source node ID (*src_nid*)
-T *secs*	Statistics mode; *secs* specifies the period (default 10 seconds).

Description:

The **netsniff** utility is an Ethernet network sniffing utility. It has been shipped as a debugging utility should you run into problems using Ethernet.

Figuring out which options to use to specify your *src_nid*/*dst_nid* filter looks a bit confusing, but really isn't. Proof by example follows:

If you want to look only at packets transmitted by node 0000c0129e40:

```
netsniff -s 0000c0129e40
```

If you want to look only at packets received by node 0000c0109e40:

```
netsniff -d 0000c0109e40
```

If you want to look only at packets transmitted by node 0000c0129e40 and received by node 0000c0109e40:

```
netsniff -s 0000c0129e40   -d 0000c0109e40
```

If you want to look only at packets transmitted or received by node 0000c0129e40 (this is a popular mode):

```
netsniff -a 0000c0129e40
```

To do anything really useful with **netsniff** on Ethernet, be sure to run your **Net.ether*** driver with the **-P** option to tell it to buffer all packets (16-bit cards are best for this). Otherwise, you will see only broadcasted packets with **netsniff**, which is usually pretty boring.

The **netsniff** utility sees only packets that aren't grabbed by some other process attaching to the raw side of things. So, QNX packets addressed to your node aren't seen, nor are TCP/IP packets if you're running **Sock[l]et**.

To see these packets, put a second ethernet card in the machine, plug it into the same LAN, but make sure nobody has **netmap** entries for it. As described above, run the **Net.ether*** driver with the **-P** option so it gets all packets, then run **netsniff** on this card. You should see everything without interfering with normal QNX networking (except that the transfer of twice the normal load could slow down the machine somewhat if the bus gets busy).

See also:

Net, **Net.***, **netinfo**, **netmap**

nettrap

© 2005, QNX Software Systems

Autodetect network card and start appropriate drivers (QNX)

Syntax:

Print command lines necessary to start drivers:

nettrap [*options***] [query]**

Start drivers:

nettrap [*options***] start**

Options:

-e *port,port*	Exclude scan in this range of I/O ports (specified in hexadecimal; no leading **0x**).
-l *lan*	("el") The first driver started should use this logical LAN number; additional drivers will be assigned LAN numbers incrementally from this value. (Default: 1)
-v	Verbose output to help debugging failed detection. Shows which cards are being scanned for.
-vv	More verbose output. Shows which cards are being scanned for, and which I/O or memory locations are being hit.
start	Start the appropriate drivers.
query	List the drivers that would be invoked by **nettrap start**. This is the default action.

Description:

The **nettrap** utility scans the machine it is run on for network cards. If **start** is not specified, the command line that would be used to start the network drivers will be printed out. If **start** is specified, **Net**, the appropriate Net drivers, and **netmap -f** will be executed.

This is ideal for the **sysinit** file on a machine booting from hard disk.

WARNING: Since the nettrap utility reads and writes to I/O ports and memory, running nettrap may cause drivers and their hardware (and not just other Net.* drivers) to behave in unpredictable ways. If sensitive or hazardous equipment is controlled by the computer, avoid using the *trap trap utilities or disconnect those peripherals before using them.

The network cards scanned for include the following hardware types:

- 3Com 503 Etherlink II Ethernet
- 3Com 509 Etherlink III Ethernet
- 3Com 905 Fast Etherlink XL
- Arcnet (Corman)
- Arcnet (standard)
- DEC FDDI
- IBM (and TROPIC) Token-Ring ISA/MCA
- IBM PCI Token-Ring
- Intel 82557/8 10/100 Ethernet
- Intel 82595 Ethernet
- Intel PRO/100 VE
- National Semiconductor DP83815 Ethernet
- NE1000/2000 Ethernet
- NE2100 (AMD 79c960/961/965/970/974) Ethernet
- SMC EtherPower (DEC 21040/21041/21140) 10/100 Ethernet
- SMC 91C90/92 Ethernet

- SMC 9432 (Epic) Etherpower II 10/100
- SMC/WD 8003/8013/elite/ultra Ethernet
- SymBios 53C885 Ethernet

See the usage message (**use nettrap**) for a definitive list of currently supported network hardware types.

The first **Net** driver started will be assigned logical LAN number one (**-l1**). The second **Net** driver will be started with **-l2**, etc. The starting logical LAN number may be made something higher than 1 by using the **-l** *lan* option.

The **nettrap** utility may be run only by **root**.

Files:

The standard input is unused.

Card information and **-v** output is written to the standard output.

In **start** mode, the standard output is unused.

Exit status:

>0	Failed to detect card and/or start driver.
0	Successfully detected card/started driver.

See also:

Net, **Net.***, **disktrap**, **mousetrap**, **pcmciatrap**, **ramtrap**

Installation & Configuration

newgrp

Change to a new group (POSIX)

Syntax:

newgrp [-|-l|-s] [*group***]**

Options:

-	Deprecated. Synonymous to **-l**.
-l	("el") Change the environment to what would be expected if the user actually logged in again.
-s	Set the group ID of the parent, do not *exec()* a login shell.
group	A group name.

Description:

The **newgrp** utility starts a new shell with a new real and effective group ID. The user remains logged in and the current directory doesn't change, but file access permissions are based on the new real and effective group IDs.

If no operands are specified, **newgrp** changes the group ID back to the group identified in the invoking user's entry in **/etc/passwd**.

If you don't use the **-l** option, the environment variables remain unchanged.

Note that group passwords aren't supported in QNX. Only the user **root** may change the group ID to a group that it doesn't belong to.

Files:

/etc/group	Contains information on the group IDs in the system. See **passwd**'s "Files" section for a brief description of the format of this file.
/etc/passwd	Contains information on the user IDs in the system.

See the **passwd** utility "Files" section for a brief description of the format of these files.

© 2005, QNX Software Systems

See also:

su, **passwd**

Syntax:

nice [-n*prioritylevels***]** *command* **[***arguments***]...**

Deprecated:

nice [-*prioritylevels***]** *command* **[***arguments***]...**

Options:

-*prioritylevels* Deprecated. This is the historical method of specifying the amount to adjust the current priority by. Specifying **-1** ("one") would lower the priority by one, while specifying **--1** ("one") would boost the priority by one.

-n *prioritylevels* Specify the amount to adjust the current priority by when running the command. The command will be run at a priority level of the current priority minus *prioritylevels*. The lowest priority allowed is priority 1. The highest for non-**root** users is 19; 29 for **root**. (To increase priority specify a negative number for *prioritylevels*.)

command [*arguments*]...

The command to run at the altered priority.

Description:

The **nice** utility invokes the specified command with a modified priority, usually making the command behave "nicer" towards competing processes.

If no *prioritylevels* option is specified, the program is invoked at a priority that is one level lower than the parent's current priority (i.e. it is invoked with a "nice increment" of 1).

If a *prioritylevels* option is specified, it is subtracted from the parent's current priority and the program is invoked at the resultant priority. If

the resulting priority is not a valid priority, **nice** will write a diagnostic message to the standard error and will exit with an exit status of 1.

If you enter:	nice will:
a positive value (e.g. **-n2** or **-n+2**)	*lower* the priority of the program, making it "nice"
a negative value (e.g. **-n-2**)	*raise* the priority of the program, making it "mean"

Users may adjust the priority as follows:

If you're:	You can change to any priority:
a normal user	from 1 to 19
root	from 1 to 29

Examples:

Run **make** at one priority lower than the parent's priority (be nice):

```
nice make application
```

Run **make** at two priorities lower than the parent's priority (be nicer):

```
nice -n2 make application
```

Run **make** at two priorities *higher* than the parent's priority (be mean):

```
nice -n-2 make application
```

Exit status:

If the operation is successful, the exit status of the invoked command is returned. If an error occurs, the exit status is as follows:

1	Invalid command-line parameters were given or the user requested an invalid priority.
126	The *command* specified did not exist.
127	The *command* could not be started for some other reason.

Caveats:

In contrast to other operating systems, the QNX interpretation of the nice value substantially affects the priority of the process. Rather than representing a *fraction* of a priority, the granularity of the nice value in QNX is of a "whole" priority level. For example, where the following would have a marginal effect on the execution of **myprog** on some operating systems:

```
nice -n5 myprog
```

on QNX it would lower the priority of **myprog** by five full priority levels, and could have a significant effect on **myprog**'s execution time.

See also:

renice, **slay**

noditto

© 2005, QNX Software Systems

Prevent another user from dittoing your machine (QNX)

Syntax:

noditto [-g *groupno***]... [-u** *userno***]...**

Options:

-g *groupid*	Allow users with this group ID to ditto.
-l *file*	("el") Specify log file to append to (implies **-v**).
-u *userid*	Allow users with this numerical user ID to ditto.
-v	Verbose, print notification of ditto attempts to **/dev/tty**.

Description:

The **noditto** utility is used to prevent other users from viewing or controlling your console screens using **ditto**. By default it disallows all ditto attempts (except those by **root**, who is immune from **noditto**). Users and members of groups may be selectively allowed by specifying one or more **-g** or **-u** options.

The attempts (both successful and unsuccessful) to ditto your machine may be logged by **noditto** by specifying the **-v** or **-l** options. The **-v** option simply logs attempts to **/dev/tty** (the console it was started from, regardless of whether the standard output has been redirected), while **-l** writes the log information to the specified filename.

The **noditto** utility forks a copy of itself in the background, then exits. The forked copy arbitrates whether **ditto** has permission to ditto the machine. To re-allow dittoing for all users, use **slay noditto**.

Examples:

Prevent all users (except **root**, who is exempt) from dittoing your machine:

```
noditto
```

Allow only users in group **techies** to ditto your machine:

```
noditto -g techies
```

Allow only users **john** and **carol** to ditto your machine:

```
noditto -u john -u carol
```

Disallow all users but group **techies** from dittoing your machine, and log all ditto activity (including unsuccessful attempts) to **/home/fred/dittolog**:

```
noditto -g techies -l /home/fred/dittolog
```

Exit status:

0 **noditto** was successfully started

>0 An error occurred (bad command-line parameter, unable to fork, unable to register name, etc.)

See also:

ditto

nohup

© 2005, QNX Software Systems

Invoke a command immune to hangups (POSIX)

Syntax:

nohup *command*

Options:

command The command to be invoked.

Description:

The **nohup** utility invokes the specified command. When the command is invoked, the SIGHUP signal is set to be ignored; thus, the command is made immune to hangups.

If the standard output is a terminal, all output written by the specified command to its standard output is appended to the **nohup.out** file in the current directory. If this **nohup.out** file can't be created, or if it can't be opened for appending, the output is appended to the end of the **nohup.out** file in the directory specified by the **HOME** environment variable. If neither **nohup.out** file can be created or opened, the specified command isn't invoked.

If the standard error output is a terminal, all output written by the specified command to its standard error output is redirected to the same file descriptor as the standard output.

It's often desirable to apply **nohup** to pipelines or lists of commands. You can do this by placing pipelines and command lists in a single file; this file can then be executed as a command, and the "nohup" applies to everything in the file.

Alternatively, the following can be used to apply **nohup** to a complex command:

```
nohup sh -c 'complex-command-line'
```

Similarly, you can use the following commands to apply **nohup** to a shell function:

```
export -f func
nohup sh -c 'command line invoking func'
```

If you want the SIGQUIT signal to also be ignored, you can run **nohup** in the background:

nohup *command* **&**

Files:

./nohup.out

If possible, this file will be created and the output of the command will be written to it.

$HOME/nohup.out

If it was not possible to open **./nohup.out**, **nohup** will attempt to create (if necessary) and append the output of the command to this file. If neither file can be opened, **nohup** will not run the command.

Environment variables:

HOME If the output file **nohup.out** cannot be created in the current directory, **nohup** uses the directory named by this variable to create the file.

PATH Contains the search path used to locate the command to be invoked.

Exit status:

The exit status of **nohup** cannot be relied upon.

on

© 2005, QNX Software Systems

Execute a command on another node or tty (QNX)

Syntax:

on [-d] [-f *nid***] [-h] [-n** *node***] [-p** *prio***] [-r** *root***] [-s] [-t** *tty***]** *command*

Options:

-d	Detach *command* from its parent (i.e. sever the parent/child relationship). This is useful for remotely created processes that will never exit and that the Shell therefore doesn't need to wait for. Unless this option is specified, a virtual circuit is created connecting the parent to the child.
-f *nid*	Spawn from remote node using remote node's **/** as prefix root. Equivalent to **-n** *node* **-r //***node***/**.
-h	Start *command* in a HELD state. This option is useful for starting up programs with the intention of debugging them. You can also start up several commands in the HELD state, then send them all a signal to start — they'll all start at almost the exact same time, since their load times will have been eliminated.
-n *node*	Execute *command* on the specified node. This is identical to the **onnode** command built into sh.
-p *prio*	Execute the command at the specified priority.
-r *root*	Use the specified default network root for *command*.
-s	Spawn the command in a new process group.
-t *tty*	Open the specified terminal name as file descriptors 0, 1, and 2 for *command*. The *command* is run in a new session with *tty* as its controlling terminal. This is identical to the **ontty** command built into **sh**.
command	The command to be executed.

Description:

The **on** utility extends the process creation abilities of the Shell (**sh**). You can start a process on a remote node, on a different controlling terminal, in a HELD state for debugging or later synchronized starting, or with a default network root that differs from the one used by its parent.

If the **-d** option is *not* specified, a virtual circuit is created as a local agent for the remote child process. When the child terminates, the parent must do a *wait()* on the virtual circuit to reap the zombie process entry for the child. If the **-d** option *is* specified, the command is detached from its parent. The parent won't be able to do a *wait()* for the child, nor will it be able to control it via signals.

By default, the command is run in the current session. The **-t** option starts a new session, which means the command won't receive a SIGHUP if the current session leader terminates.

CAUTION: The **on -t** command will become the new session leader on the tty specified, i.e. it will receive SIGHUP generated by hangups on that tty. Any processes originally running on that tty will NOT get SIGHUP, and this condition will persist even when the process started by **on** has terminated. For this reason, **on -t** should specify only ttys that are not currently in use.

Examples:

Run **login** on console 2:

```
on -t /dev/con2 login
```

Run **who** on node 3:

```
on -n 3 who
```

Run **sort** as an orphan on node 3:

```
on -d -n 3 sort file.dat
```

© 2005, QNX Software Systems

Run **who** on node 7 with a new session, its standard I/O connected to console 1 on node 3:

```
on -t //3/dev/con1 -n7 who
```

Run **make** on node 3, using node 3 as the network root — this eliminates the need to resolve pathnames through the invoking node:

```
on -n 3 -r //3/ make
```

Exit status:

The **on** utility exits with the exit status of the command.

See also:

sh

passwd

Change login password (UNIX)

Syntax:

passwd [*name***]**

Options:

name The username whose password is to be changed or for whom an account is to be created (**root** only).

Description:

The **passwd** utility allows a login password to be changed or a new username to be created.

If you are changing your password, **passwd** prompts for the old password and then for the new password. The new password must be entered twice, to avoid typing mistakes. Only the owner or the superuser may change a password.

The superuser (**root**) may use this utility to create a new user account by entering:

passwd *new_user_name*

When creating a new user account, **passwd** will prompt for information, such as the user's group list, home directory, and shell.

The **/etc/default/passwd** file lets you modify **passwd**'s behavior to suit local requirements.

The **/etc/passwd** file contains the following fields, separated by colons:

username:*has_passwd*:*userid*:*groupid*:*misc*:*home_directory*:*initial_command*

If the *has_passwd* field contains an **x** character, a password has been defined for this user. If no character is present, no password has been defined. Use of any other character is reserved and may cause side-effects for the user.

The *groupid* field contains a group number. Users may log in under the *groupid* listed in their **/etc/passwd** file entry without being listed as a member of that group in the **/etc/group** file.

The *misc* field stores supplemental information, with commas separating subfields. Usually, the first subfield contains the user's "real life" name. Some utilities use this information (e.g. **finger**).

The *initial_command* field contains the initial command to run after the user has successfully logged in. This command and any arguments it takes must be separated by tab or space characters. As the command is spawned directly (not run by a shell), no shell expansions will be performed. There is no mechanism for specifying command-line arguments that contain space or tab characters themselves. (Quoting is not supported.)

If no *initial_command* is specified, **/bin/sh** will be used.

Files:

/etc/.pwlock This file is created by **passwd** to indicate to other instances of **passwd** that the password file is currently being modified. When **passwd** finishes, the file is removed. See "Caveats," below.

/etc/group This file defines the known groups for the system. It associates group *names* with a numerical *ID* and a list of *username*s who are members of the group.

Entries in this file appear in the following format:

groupname::*groupid*:*user*[,*user*]...

/etc/opasswd
/etc/oshadow

When **passwd** modifies a password file, it first locks the password files with the **/etc/.pwlock** file, then copies the contents of the current **/etc/passwd** and **/etc/shadow** files to **/etc/opasswd** and **/etc/oshadow**, respectively. If **passwd** is killed before it finishes writing the

updated file, the password files may be restored from these backup versions. See "Caveats," below.

/etc/default/passwd

Contains configuration variables for the **passwd** utility. At least the following items may be set in this file:

BASEDIR=*dirname*

This specifies the default base directory under which user directories will be created. (Default: **/home**)

SHELL=*progname*

Specifies the default shell to be used for the *initial_command* field in new password entries. (Default: **/bin/sh**)

UIDRANGE=*low*-[*high*]

Specifies the range of values which may be selected for new new user IDs. The *high* component of the range may be omitted, indicating no upper bound. The dash (**-**) is still required in this case. (Default: 100-)

GIDRANGE Specifies the range of values which may be selected for group IDs. As with **UIDRANGE**, the absence of a *high* component indicates there is no upper bound. (Default: 100-)

DUPDIROK If specified, **passwd** will allow the selection of an already existing directory as a new user's home directory. (Default: not allowed)

© 2005, QNX Software Systems

DUPUIDOK If specified, **passwd** will allow the selection of an already existing user ID for a new user name. This is generally discouraged since it would mean allowing a many-to-one user name/ user ID relationship. (Default: not allowed)

NOPASSWORDOK

If specified, **passwd** will allow user accounts to be set up which do not require a password to log in. (Default: not allowed)

STRICTPASSWORD

If specified, **passwd** will require all passwords to contain at least two types of characters (e.g. alphabetic and punctuation) (Default: strict passwords not required)

INSISTANT=*retries*

Specifies the number of times **passwd** will ask a non-**root** user if they really want to set their account up with no password. Ignored if **NOPASSWORDOK** is set.

PROFILE=*basename*

Specifies the basename of the file in the user's home directory which will be used as the shell initialization file. The **DEFPROFILE** (below) file will be copied there when a new account is set up. (Default PROFILE=**.profile**)

DEFPROFILE=*filename*

Specifies the pathname to a default shell initialization file which will be copied to a new user's **PROFILE** when their account is set up.

/etc/passwd Contains the user account entries. The format of entries in this file are as follows:

username:*has_passwd*:*userid*:*groupid*:*misc*:*home_directory*:*initial_command*

/etc/shadow Contains encoded versions of the actual passwords for user accounts. The passwords themselves are not stored in the **/etc/passwd** file.

Caveats:

The **passwd** utility creates the **/etc/.pwlock** file during updates to the password database. If for some reason the system crashes at an inopportune moment and leaves this file present, **passwd** will refuse to work until the file is removed by the system administrator. If the password files are somehow left in an inconsistent state as a result of the crash, the system administrator should also copy **/etc/oshadow** to **/etc/shadow** and copy **/etc/opasswd** to **/etc/passwd**.

See also:

login, **su**

QNX Installation & Configuration

paste

© 2005, QNX Software Systems

Merge corresponding or subsequent lines of files (POSIX)

Syntax:

paste [-d *list***] [-s]** *file***...**

Options:

-d *list* Use one or more characters specified in *list* to separate corresponding lines in the output (default delimiter is **tab**). If *list* contains multiple characters, the characters are used circularly; i.e. when the list is exhausted, the first character from the list is reused. You can use the following special characters in *list*:

Character	Represents
\n	**newline** character
\t	**tab** character
****	**backslash** character
\0	empty string (not a null character)

In parallel merging (no **-s** option), the lines from the last file always end with a **newline** character instead of the one specified in *list*.

-s Merge the subsequent lines of each separate input file into a single line. When **paste** finishes merging all the lines in one file, it forces a new line and then merges the lines of the next file. A **tab** separates the merged lines unless you specify another character with option **-d**. Regardless of the *list* used with **-d**, the last character of the file is always a **newline** character.

file The pathname of a text file, whose contents are used instead of the standard input. If a dash (**-**) is specified for one or more of the filenames, the standard input is used;

the standard input is read one line at a time, round-robin, for each instance of a dash (**-**).

Description:

The **paste** utility reads input files, concatenates their corresponding lines, and writes the resulting lines to the standard output.

By default, **paste** treats each file as a column and places the columns side by side. This is known as "parallel merging." If **-s** is specified, however, **paste** combines the subsequent lines of each input file into a single line. This is known as "serial merging."

Output lines are separated by the **tab** character unless another delimiter is specified by option **-d**.

Examples:

List a directory in one column:

```
ls -C | paste -d" " -
```

List a directory in four columns:

```
ls | paste - - - -
```

Combine pairs of lines from **myfile** into single lines, separated by a **tab**:

```
paste -s -d "\t\n" myfile
```

The following examples show how **paste** operates on two simple files, each containing four lines:

myfile contains the following:

```
fred
barney
wilma
dino
```

© 2005, QNX Software Systems

yourfile contains the following:

```
george
judy
jane
astro
```

Executing **paste myfile yourfile** results in:

```
fred    george
barney  judy
wilma   jane
dino    astro
```

Executing **paste -s myfile yourfile** results in:

```
fred    barney  wilma    dino
george  judy    jane     astro
```

Files:

The input files are text files.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

cat, **cut**, **grep**, **join**

Syntax:

List archive contents:

pax [-cimopuvy] [-f *archive***] [-s** *replstr***]... [-t** *device***]**
[*pattern***...]**

Read an archive:

pax -r [-cimnopuvy] [-f *archive***] [-s** *replstr***]...**
[-t *device***] [***pattern***...]**

Write an archive:

pax -w [-dimuvy] [-b *blocking***] [-[a]f** *archive***]**
[-s *replstr***]... [-t** *device***] [-x** *format***] [***pathname***...]**

Copy files:

pax -rw [-ilmopuvy] [-s *replstr***]... [***pathname***...]** *directory*

Options:

-a Append the files specified by *pathname* to the archive specified with **-f**.

-b *blocking* Block the output at *blocking* bytes per write to the archive file. A **k** suffix multiplies blocking by 1024, a **b** suffix multiplies blocking by 512, and an **m** suffix multiplies blocking by 1048576 (1 megabyte). If not specified, blocking is automatically determined on input and is ignored for **-rw** (copy files).

-c Complement the match sense of the *pattern* operands.

-d Don’t create intermediate directories not explicitly listed in the archive. This option is applied only if the **-r** option is specified.

-f *archive* Use *archive* as the pathname of the input or output archive, overriding the default of standard input for **-r** or standard output for **-w**.

-i Interactively rename files. Substitutions specified by **-s** options are performed before requesting the new filename from the user. A file is skipped if an empty line is entered. If EOF is encountered, **pax** exits with an exit status of 0.

-l ("el") When possible, link rather than copy files.

-m Don't keep file modification times.

-n When **-r** is specified, but **-w** is not, treat the *pattern* operands as ordinary filenames. Only the first occurrence of each of these files in the input archive is read. The **pax** utility exits with an exit status of 0 after all files in the list have been read. If one or more of the files in the list aren't found, **pax** writes a diagnostic to standard error for each of these files and exits with a nonzero exit status. The filenames are compared before any of the **-i**, **-s**, or **-y** options are applied.

-o Restore file ownership as specified in the archive. The invoking process must have appropriate privileges to accomplish this.

-p Preserve the access time of the input files after they have been copied.

-s *replstr* Modify filenames according to the substitution expression. The syntax for the expression is:

-s /*old*/*new*/[**gp**]

Any non-null character may be used as a delimiter (a **/** is used here as an example). Multiple **-s** expressions are applied in the order specified

terminating with the first successful substitution. The optional trailing **p** causes successful mappings to be listed on standard error. The optional trailing **g** causes the old expression to be replaced each time it occurs in the source string. Files that substitute to an empty string are ignored both on input and output.

-t *device* The *device* argument names the input or output archive device, overriding the default of standard input for **-r** and standard output for **-w**.

-u Copy each file only if it is newer than a preexisting file with the same name.

-v Be verbose; list filenames as they are encountered. This option produces a table of contents listing on the standard output when both **-r** and **-w** are omitted; otherwise the filenames are printed to standard error as they are encountered in the archive.

-x *format* Use this output archive format. The input format, which must be one of the following, is automatically determined when the **-r** option is used. The supported formats are:

- **cpio** The extended **cpio** interchange format specified in POSIX Std 1003.1-1988.
- **ustar** The extended **tar** interchange format, also specified in POSIX Std 1003.1-1988. This is the default archive format.

-y Interactively prompt for the disposition of each file. Substitutions specified by **-s** options (described above) are performed before the user is prompted for disposition. EOF or an input line starting with the character **q** causes **pax** to exit. Otherwise, an input line starting with anything other than **y** causes the file to be ignored. This option cannot be used in conjunction with the **-i** option.

Only the last **-f** or **-t** option takes effect.

directory The destination directory pathname for copies when both the **-r** and **-w** options are specified. The directory must exist and you must have the appropriate write permissions, or an error results.

pathname A file to be copied or a directory containing files and subdirectories to be (recursively) copied in addition to the directory itself.

pattern A pattern given in the standard shell pattern-matching notation. If no pattern is specified, the default is *****, which selects all files.

Modes of operation:

If you don't specify **-r** or **-w**, then **pax** lists the contents of the specified archive. In this mode, **pax** lists normal files one per line. Hard link pathnames are listed as:

pathname == *linkname*

where *pathname* is the name of the file being extracted, and *linkname* is the name of a file that appeared earlier in the archive.

Symbolic link pathnames are listed as:

pathname -> *destination_path*

If **-v** is specified, then **pax** lists normal pathnames in the same format used by the **ls** utility with the **-l** ("el") option, except for hard links which are shown as:

<ls -l *listing***>** == *linkname*

The modes of operation related to combinations of **-r** and **-w** are as follows:

-r Read an archive file from the standard input; select for extraction only those files whose names match any of the *pattern* operands. The selected files are conditionally created and copied relative to the current directory tree, subject to the options chosen. By default, the owner and group of selected files will be that of the invoking process, and the permissions and modification times will be the same as those in the archive.

The supported archive formats are automatically detected on input.

-w Write the files and directories specified by *pathname* operands to the standard output, together with the pathname and status information prescribed by the archive format used. The default output format is **tar**, but may be overridden by the **-x** format option described below.

A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, then the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames appearing on the standard input are copied.

-rw Read the files and directories named in the *pathname* operands and copy them to the destination directory. A directory *pathname* operand refers to the files and (recursively) subdirectories of that directory. If no *pathname* operands are given, the standard input is read to get a list of pathnames to copy, one pathname per line. In this case, only those pathnames appearing on the standard input are copied. The directory named by the directory operand must exist and must have the proper permissions before the copy can occur.

Description:

The **pax** utility reads and writes archive files that conform to the archive/interchange file format specified in POSIX Std 1003.1-1988.

The utility can also read, but not write, a number of other file formats. Support for these traditional file formats (such as V7 **tar** and System V binary **cpio** format archives) is provided for backward compatibility and to maximize portability.

The **pax** utility will also support traditional **cpio** and System V **tar** interfaces if invoked with the name "**cpio**" or "**tar**" respectively (these are links to **pax**).

The **pax** utility is capable of reading and writing archives that span multiple physical volumes. Upon detecting an end of medium on an archive that is not yet completed, **pax** will prompt you for the next volume of the archive and will let you specify the location of the next volume.

If the pax archive is stored directly in a floppy disk block special file (e.g. **/dev/fd0**), the archive will have overwritten the first block of the disk, which the QNX floppy driver, **Fsys.floppy**, uses to store media type information that lets it dynamically adjust to diskettes of differing media capacity being inserted in the drive (e.g. 720k vs 1.4M, 360k vs 1.2M etc.). It is therefore recommended that for both reading or writing archives directly to a floppy drive, you either pipe the archive through **vol** which skips this first block, or lock the floppy drive to a specific media type using **lockfd** which causes the driver to stop looking at the first block media descriptor.

Combinations of the **-r** and **-w** command-line arguments specify whether **pax** will read, write, or list the contents of the specified archive, or will move the specified files to another directory.

When writing to an archive, the standard input is used as a list of pathnames if no *pathname* operands are specified. The format is one pathname per line. When reading, the standard input is the archive file, which is formatted according to one of the format specifications in POSIX Std 1003.1-1988.

The user ID and group ID of the process, together with the appropriate privileges, affect the ability of **pax** to restore ownership

and permissions attributes of the archived files. (See Archive/Interchange File Format in POSIX Std 1003.1-1988.)

Note that the options **-a**, **-c**, **-d**, **-i**, **-l**, **-p**, **-t**, and **-y** are provided for functional compatibility with the historical **cpio** and **tar** utilities. The option defaults were chosen based on the most common usage of these options, so some of the options have meanings different from those of the historical commands.

Examples:

Copy the contents of the current directory to the floppy drive:

```
pax -w -f /dev/fd0 .
```

Copy the contents of **olddir** to **newdir**:

```
mkdir newdir
cd olddir
pax -rw . ../newdir
```

Read the archive **pax.out** with all files rooted in **/usr** in the archive extracted relative to the current directory (note the use of commas as pattern separators for the **-s** option):

```
pax -r -s ",^/usr/,," -f pax.out
```

Files:

The controlling terminal (**/dev/tty**) is used to prompt the user for information when the **-i** or **-y** options are specified.

Exit status:

0 All files in the archive were processed successfully.

>0 The **pax** utility aborted due to errors encountered during operation.

Caveats:

Special permissions may be required to copy or extract special files.

Device, user ID, and group ID numbers larger than 65535 cause additional header records to be output. These records are ignored by some historical versions of **cpio** and **tar**.

The archive formats described in Archive/Interchange File Format have certain restrictions that have been carried over from historical usage. For example, pathnames stored in the archive can be no more than 255 characters in length.

When getting an **ls -l** style listing on **tar** format archives, link counts are listed as zero since the **ustar** archive format doesn't keep link count information.

On 16-bit architectures, including 16-bit versions of QNX 4., the largest buffer size is 32K-1. This is due, in part, to using integers in the buffer allocation schemes. On many of these machines, however, it is not possible to allocate blocks of memory larger than 32K.

See also:

cp, **cpio**, **find**, **lockfd**, **tar**

You must be **root** to run this utility.

Syntax:

pc-pin [-s *socket***] [***command***]**

Options:

-s *socket* Display information about this PC Card socket only (starting at 1).

command The type of information to display:

Command	Description
status	Show the status of each socket (default)
config	Attempt to make configuration file entries (obsolete)
cis	Decode the Card Information Structure (CIS) on the card

Description:

The **pc-pin** utility displays useful information about PC Card resources, servers, and cards.

The QNX 4 PC Card server (**pccard**) automatically configures PCMCIA/CardBus I/O cards as they're inserted. It also manages computer resources such as memory windows, ports, and IRQs. Resources are allocated as cards are inserted, and then freed as cards are removed.

Drivers that are PC Card-aware can be informed of card insertions and removals, based on the type of card; these drivers can then configure themselves to use the correct ports and IRQs.

© 2005, QNX Software Systems

The **pccard-launch** utility automatically starts drivers and passes them command-line arguments describing the ports and IRQs used by the card, thus permitting a non-PC Card-aware driver to be automatically started when the card is inserted, and stopped when it is removed.

Exit status:

>0	Failed
0	Success

See also:

pccard, **pccard-launch**

You must be **root** to start this driver.

Syntax:

pccard [*options***]...**

Options:

-a The ioport address to be assigned to a PCMCIA card. Use a colon (:) to separate functions on multi-function cards. E.g. **-a 0x300:0x320,0x340** will assign ioport **0x300** to function 1 in socket 0 and ioport **0x320** to function 2 in socket 0; ioport **0x340** will be assigned to function 1 in socket 1.

-i *irq* The interrupt to use for insertion/removal events. The default is no interrupts — the adapter is polled every second for status changes (recommended).

-l ("el") Override PCMCIA IRQ for socket(s). E.g. **-l5** will assign IRQ 5 to the card in socket 0; **-l5,7** will assign IRQ5 to the card in socket 0 and IRQ 7 to the card in socket 1.

-v Be verbose, for debugging purposes.

-w Force the width for a PCMCIA socket (8 or 16 bits). E.g. **-w8** will force an 8-bit width for socket 0; **-w16,8** will force a 16-bit width for socket 0 and an 8-bit width for socket 1. This option might be needed for certain PCMCIA Ethernet adapters that report themselves as 16-bit, but only work in 8-bit mode.

Description:

The **pccard** server provides support under QNX 4 for PCMCIA and CardBus host adapter chips. The host adapters currently supported are (PCMCIA) Intel 82365, Cirrus CL-PD67*xx*, Vadem VG-46*x*, (CardBus) TI-11*xx*, TI-12*xx*, and TI-14*xx*, Ricoh R5C47*x*, O2 Micro OZ68*xx*, and Toshiba Topic97. Other CardBus adapters work only in legacy (PCMCIA) mode.

The server manages host resources (memory windows, I/O ports and IRQs) and assigns resources to PCMCIA as well as CardBus cards as they are inserted. The **pccard** server also supports dual-function PC Cards and assigns separate resources to each function. The only common resource assigned to dual-function PC Cards is the IRQ.

Utilities are provided to start and stop processes (as cards are inserted and removed), display server status, and display card CIS (Card Information Structure) data.

The executables involved in PC Card support are:

pccard The server for PCMCIA and CardBus adapters.

pccard-launch
A manager that starts and stops processes as cards are inserted and removed.

pc-pin A utility that displays PC Card information (CIS, status, and so on).

Resources and Server Configuration Files

The server manages separate resource pools for memory windows, IRQs, and ports. When a card is inserted, the server attempts to satisfy the card's memory window, IRQ, and port requirements by allocating resources from the various pools. PCMCIA resources must be in the ISA range of devices, while CardBus resources must be in the PCI range. PC Card resource pools are created as described below.

PCMCIA

Upon startup the **pccard** server will attempt to read the **/etc/config/pcmcia.node** file, where **node** is replaced by the node number of the machine the server is running on. If this file does not exist, the resources will be read from the **/etc/config.pcmcia.cfg** file. This file is identical to the one used by **Pcmcia.generic**, so it is suggested that you keep your original file.

CardBus

CardBus resources are assigned by the **pccard** server. When the server starts, it scans the PCI bus and excludes all resources that have already been assigned to devices on the PCI bus by the BIOS. When a CardBus card is inserted in a socket, **pccard** will determine the memory and I/O port requirements of the card and assign the required resources to the card. The IRQ will be the same as that assigned to the CardBus adapter by the BIOS.

See also:

pccard-launch, **pc-pin**

pccard-launch

© 2005, QNX Software Systems

Automatically start drivers when PC Cards are inserted

You must be **root** to run this utility.

Syntax:

pccard-launch [-nv] '*type*,*command*'...

Options:

-n Don't lock the socket where the expected card is found.

-v Display verbose messages on standard output.

'*type,command*'

For this PC Card type, spawn *command*. Types may be specified in decimal (*nnnn*) or hexadecimal (**0x***nnn*), and are followed by a comma (**,**) and the command to execute. For example:

```
pccard-launch '0x600,Net.ether1000 ioport=0x$IOPORT,irq=$IRQ' &
```

You can specify multiple type/command pairs on the command line by separating them with whitespace. If the command specified contains whitespace, the command must be enclosed in quotes (as shown).

Description:

The **pccard-launch** utility automatically starts drivers and sets environment variables describing the ports and IRQs used by a card, thus permitting a non-PC Card-aware driver to be started automatically when a card is inserted, and stopped when it is removed.

The environment variables may be named in the command lines for the drivers to be run, thus allowing the port and IRQ information to be passed on the command line to the drivers, most of which aren't aware of the environment variables set by **pccard-launch**.

This utility both starts a driver when a card is plugged in and stops it, via a SIGTERM signal, when the card is removed.

When a card is plugged in, **pccard-launch** checks the card type against the list of card types/driver commands supplied to it on its command line. If it doesn't know about the card type, it does nothing. If it has been provided with a command for the card type, it does the following:

- Locks the card.
- Sets up environment variables detailing I/O port locations, IRQ used, etc.
- Spawns the associated command.

When the card is removed, **pccard-launch** unlocks the card and signals the previously spawned process with SIGTERM.

You can determine the card type by using the **pc-pin** command. A **pc-pin con** command displays configuration information for the PC Card. Look at the **config** line, which should look something like:

config = 0x*NN*, **0x***TTTT*...**"**

where *NN* is the configuration number and *TTTT* is the type of the card. For example, a modem card has a type of **0x200**. This is the type that you must use on the command line for **pccard-launch**.

Examples:

Automatically start the Net.ether1000 driver with the appropriate I/O port and IRQ settings, for PC Card type 0x600:

```
pccard-launch '0x600,Net.ether1000 ioport=0x$IOPORT,irq=$IRQ' &
```

© 2005, QNX Software Systems

Files:

pccard-launch

This utility is located in the **/bin/** directory.

Environment variables:

The environment variables set by **pccard-launch** are:

IOPORT	Hex address of the I/O port (e.g. 320).
IOPORTSZ	Size of the I/O port (e.g. 32).
IOPORT2	Hex address of the second I/O port, if assigned.
IOPORT2SZ	Size of the second I/O port, if assigned.
IRQ	IRQ of the device.
SOCKET	The socket where the card is inserted.

See also:

pccard, **pc-pin**

PcmciaBeep

Beep the PC speaker when PCMCIA cards are changed (QNX)

Syntax:

```
PcmciaBeep &
```

Options:

None.

Description:

The **PcmciaBeep** utility runs in the background and beeps the PC's speaker whenever a PCMCIA card is changed.

The sounds made by **PcmciaBeep** vary depending on what the change was:

Beeps	Description
LOW,HIGH	Card configured and in use by a device driver
LOW	Card configured but no device driver attached
LOW,RAZZ	Card recognized, but resources unavailable to configure
HIGH,LOW	Card has been removed
RAZZ	Card is not recognized
HIGH,HIGH,HIGH	Card battery low

Examples:

Automatically beep the speaker whenever a PCMCIA card is inserted or removed:

```
PcmciaBeep &
```

© 2005, QNX Software Systems

Exit status:

The **PcmciaBeep** utility does not normally exit. If supplied with invalid command-line parameters, or if it cannot find a PCMCIA server, it will write a diagnostic message to the standard error and will exit with a non-zero exit status.

See also:

Pcmcia.*, **PcmciaLaunch**, **pcmciatrap**, **pin**

Automatically start drivers when PCMCIA cards are plugged in (QNX)

Syntax:

PcmciaLaunch [-v] '*type*,*command*'...

Options:

-v Display verbose messages to the standard output.

'*type,command*'

For this PCMCIA card type, spawn the *command*. Types may be specified in decimal (*nnnn*) or hexadecimal (**0x***nnn*), and are followed by a comma (**,**) and the command to execute. e.g. **'0x601,Net.ether1000 -p$IOPORT -i$IRQ'**. Multiple type/command pairs may be specified on the command line separated by white space. If the command specified contains white space, the command must be enclosed in quotes (as shown).

Description:

The **PcmciaLaunch** utility automatically starts drivers and sets environment variables describing the ports and IRQs used by a card, thus permitting a non-PCMCIA-aware driver to be automatically started when a card is inserted and stopped when removed. The environment variables may be named in the command lines for the drivers to be run, thus allowing the port and IRQ information to be passed on the command line to the drivers, most of which will not be aware of the environment variables set by **PcmciaLaunch**.

PcmciaLaunch will both start drivers when cards are plugged in and stop them via SIGTERM when cards are removed.

When a card is plugged in, **PcmciaLaunch** will check the card type against the list of card types/driver commands supplied to it on its command line. If it does not know about the card type, it does nothing. If it has been provided with a command for the card type, it does the following:

- locks the card
- sets up environment variables detailing I/O port locations, IRQ used, etc.
- spawns the associated command

When the card is removed, **PcmciaLaunch** unlocks the card and signals the previously spawned process with SIGTERM. The environment variables set by **PcmciaLaunch** are:

Envar	Specifies
IOPORT	Hex address of I/O port (e.g. 320)
IOPORTSZ	Size of I/O port (e.g. 32)
IOPORT2	Hex address of second I/O port if assigned
IOPORT2SZ	Size of second I/O port if assigned
IRQ	IRQ of device
DMA	DMA level if assigned
SOCKET	The PCMCIA socket where the card is inserted

Examples:

Automatically start the **Net.ether1000** driver with the appropriate I/O port and IRQ settings, for PCMCIA card type 0x601:

```
PcmciaLaunch '0x601,Net.ether1000 -p$IOPORT -i$IRQ'
```

Exit status:

The **PcmciaLaunch** manager does not normally exit. If supplied with invalid command-line parameters, it will write a diagnostic message to the standard error and will exit with a non-zero exit status.

See also:

Pcmcia.*, **pcmciatrap**, **pin**

Pcmcia.*

© 2005, QNX Software Systems

PCMCIA servers for various chipsets (QNX)

Syntax:

Pcmcia.generic [*options***]...**

Pcmcia.365sl [*options***]...**

Pcmcia.cirrus [*options***]...**

Options:

-3 Use this option if you have a PD6729 on a PCI bus that's ≥ 33MHz.

-A *index* For AMD SC400 in Standard Mode, specify the value to be written to the PC Card Extended Features Register to redirect memory windows from Socket A to Socket B. Here are the possible values and corresponding allocation of windows for Socket A and B:

Value	Socket A	Socket B
0	0 1 2 3 4	0
1	0 2 3 4	0 1
2	0 1 3 4	0 2
3	0 3 4	0 1 2
4	0 1 2 4	0 3
5	0 2 4	0 1 3
6	0 1 4	0 2 3
7	0 4	0-3
8	0 1 2 3	0 4
9	0 2 3	0 1 4

continued...

Value	Socket A	Socket B
a	0 1 3	0 2 4
b	0 3	0 1 2 4
c	0 1 2	0 3 4
d	0 2	0 1 3 4
e	0 1	0 2 3 4
f	0	0 1 2 3 4

-a *address*
The I/O port address of the PCMCIA adapter chip, in hex. The default is 0x3E0 for **Pcmcia.generic**, **Pcmcia.cirrus**, and **Pcmcia.365sl**.

-b
The system was booted from a PCMCIA memory card. This tells the server to treat enabled memory windows as boot partitions (implies **-R**).

-c "*config***"**
Specify configuration directly on command line. The configuration format is identical to the configuration file format, with the exception that lines are separated by semicolons (**;**).

-d
Tell the server to use DMA (if the card supports it).

-F *filename*
Read resources and card descriptions from this file. If this option isn't specified, the server will read from **/etc/config/pcmcia.cards** and then either **/etc/config/pcmcia.***node* or **/etc/config/pcmcia.cfg**.

-i *irq*
Use this IRQ for adapter status changes. If this flag isn't supplied, the server will allocate an IRQ from the resource pool. If you specify **-i0**, then status changes are detected by polling — no IRQ is used. This option is useful in cases where few IRQs are available for PCMCIA cards (e.g. on several laptop models).

© 2005, QNX Software Systems

-I *bitmap* Set the routable card interrupt IRQ mask. This tells the server which host interrupts card interrupts may be programmed to be. The mask is in the same format as the **-S** switch.

-M*n*:*bitmap* Specify a map of available memory windows per socket in the default or previously specified adapter. The *n* variable is the socket number. The hexadecimal *bitmap* has one bit set for each available window in the adapter. Note that only the specified windows will be reset.

-m *physaddr* Use this physical memory window (of 4K) for server usage. The address is supplied as a hex physical address. If this flag is not supplied the server will allocate a window from the resource pool.

-n *sockets* Set the number of sockets that the server thinks are available. If no **-n** option is supplied the server will attempt to determine the number of sockets based upon the PCMCIA adapter chip type.

-P Partially reset socket memory windows, preserving the ones that are already enabled at startup (for AMD SC400).

-r *delay* Delay this many milliseconds before accessing a card after reset. The default value of 500 should be sufficient for most cards, but may need to be increased for cards that take a long time to reset.

-R *socket* Forces the server to NOT reset the adapter chip upon startup. This is useful if the BIOS sets up special modes within the adapter chip, and you don't wish the defaults to be restored. Note that you can specify a particular socket (**1**, **2**, . . .).

-S *bitmap* Set the routable status IRQ mask. This tells the server which interrupts the status change output of the adapter chip can be programmed to be. Each

server has a default that corresponds to the particular chip, but certain implementations may not support all of the possible values. The value supplied is a hex number with each bit representing a single IRQ, if the bit is set then that IRQ can be used. This table shows the bit/IRQ correspondence:

Bit	IRQ
15	15
14	14
⋮	⋮
1	1
0	unused

Note that this mask tells the server only which interrupts the hardware is connected to; the interrupt must still be present in the resource pool in order for the server to use it. For example, if your adapter supported only IRQs 12, 10, 7, and 5 you would use: **`-S 14A0`**.

`-s` *size* Specify the physical memory window size (default is 4096).

`-w` Don't insert wait states on 16-bit memory accesses.

Description:

The **`Pcmcia.*`** servers support PCMCIA I/O cards through a number of different host adapter chips, including the Intel 82365 and compatibles, and the Cirrus CL-PD67xx series. The servers also support Ricoh CardBus chips.

The servers manage host resources (memory windows, ports and IRQs) and will assign resources to PCMCIA cards as they are

© 2005, QNX Software Systems

inserted. Utilities are provided to start and stop processes (as cards are inserted and removed), display server status, display card CIS (Card Information Structure) data, and automatically generate configuration entries for new cards.

The executables involved in PCMCIA support are:

Pcmcia.generic server

(PCMCIA server supporting both Intel 82365SL and Cirrus CL-PD67xx)

Pcmcia.365sl server

(Link to **Pcmcia.generic**. PCMCIA server for Intel 82365SL)

Pcmcia.cirrus server

(Link to **Pcmcia.generic**. PCMCIA server for Cirrus CL-PD67xx)

PcmciaLaunch manager

(Start processes as cards are inserted)

pcmciatrap utility

(Detect PCMCIA adapter and start appropriate server)

pin utility (PCMCIA information — CIS, cards, etc.)

PcmciaBeep utility

(Beep the PC speaker when PCMCIA cards are changed)

Selecting a Server

Servers are provided for several different PCMCIA adapter chips. The various servers and the chips they support are listed below:

Use:	For:
Pcmcia.generic	Intel 82365SL and 82365SL DF; Vadem VG-365, VG-465, VG-468 and VG-469; Cirrus CL-PD6710, CL-PD6720, CL-PD6722 and CL-PD6929; Texas Instruments PCI1131 family; AMD SC400.
Pcmcia.365sl	Intel 82365SL and 82365SL DF and Vadem VG-365, VG-465, VG-468 and VG-469.
Pcmcia.cirrus	Cirrus CL-PD6710, CL-PD6720 and CL-PD6722.

You can use the **pcmciatrap** utility to automatically start the correct server.

WARNING: The pcmciatrap utility determines the chip type by probing I/O ports that are likely to be associated with a PCMCIA adapter. If you have other adapters at these addresses the utility may confuse the hardware and/or any associated drivers. Some NE2000 Ethernet adapters can crash the machine if they are accessed improperly.

Resources and Server Configuration Files

The server manages separate resource pools for memory windows, IRQs and ports. When a card is installed the server will attempt to satisfy the card's memory window, IRQ and port requirements by allocating resources from the various pools. If insufficient resources are available the server will be unable to configure the card. There is often a need to express to the server that a particular resource be used only as a last resort or if specifically requested, therefore the server allows you to tag specific resources as reserved. For instance, some PCMCIA modem cards can only be configured to use ports 0x3f8, 0x2f8, 0x3e8 and 0x2e8. If your machine already has two serial ports installed it is likely that only 0x3e8 and 0x2e8 are available for use by the modem. If your machine had several PCMCIA sockets available

© 2005, QNX Software Systems

you would not want a card that can be configured to use any available port to use the few ports available for modem cards. In this case you would tag ports 0x3e8 and 0x2e8 as reserved; they would then be used only if a particular card (such as the modem) specifically requested them or if no other ports were available.

The PCMCIA server needs to know what resources (memory windows, IRQs and ports) are available for use. Upon startup the server will attempt to read **/etc/config/pcmcia.***node*, where *node* is replaced by the node number of the machine the server is running on. If this file does not exist, then the resources will be read from **/etc/config/pcmcia.cfg**. A default configuration file is supplied that has resource entries that may or may not match your particular environment. You should copy the default file to the node specific file and then modify it to suit your machine.

The resource section of the configuration file must start with the line:

```
[resources]
```

and must be followed by any number of the following lines:

```
rports  = reserved_port_list
ports   = normal_port_list
rirqs   = reserved_irq_list
irqs    = normal_irq_list
rmemory = reserved_memory_window_list
memory  = normal_memory_window_list
```

Each resource list consists of a number of ranges separated by commas. Ranges can appear in several different forms:

start-end	specifies a range of values including the *start* and *end* values.
start+length	specifies a range starting at *start* and continuing for *length* units.
start	specifies a range of length 1 starting at *start*.

The *start*, *end* and *length* components can be specified in decimal (start with **1** to **9**), hexadecimal (start with **0x**), or octal (start with **0**). The numbers can be followed by **k** or **M** to indicate kilobytes or megabytes. Memory ranges are specified as true physical addresses (NOT paragraphs or pages) and must be aligned on 4K boundaries.

Comments may appear anywhere and must be preceded by the **;** or **#** character, the rest of the line is ignored. Blank lines and spaces are ignored.

Note that most adapters can utilize any memory window within the 16M ISA-bus address space. For example on a machine with 8M of RAM the line:

```
memory = 9M+1M
```

would describe a 1M memory window starting at the 9M boundary. For machines with 16M of RAM installed the memory windows will most likely be restricted to be within the 640K to 1M area.

All adapters are restricted as to which IRQs they can use. For example, the Intel-based adapters can use only IRQs 15, 14, 12, 11, 10, 9, 7, 5, 4, and 3. In this case there would be no restriction on entering IRQ 6 into the resource pool (if it is available), but the server would never use it.

The server itself requires a 4K memory window and a single IRQ, your resource section must include at least these two resources.

Here is an example configuration file:

```
[resources]
memory = 0xd4000-0xd5fff
ports = 0x320-0x33f
rports = 0x2e8+8, 0x3e8+8, 0x170+8, 0x376+2
irqs = 5,7,11,12
```

© 2005, QNX Software Systems

Card Configuration Database — /etc/config/pcmcia.cards

The QNX PCMCIA server will automatically configure PCMCIA I/O cards as they are inserted. It will also manage computer resources such as memory windows, ports and IRQs. Resources will be allocated as cards are inserted and then freed as cards are removed. Drivers that are PCMCIA-aware can be informed of card insertions and removals based upon the type of card. These drivers can then configure themselves to use the correct ports and IRQs. The **PcmciaLaunch** manager will automatically start drivers and pass them command-line arguments describing the ports and IRQs used by a card, thus permitting a non-PCMCIA-aware driver to be automatically started when a card is inserted and stopped when removed.

Each I/O card must contain a CIS (Card Information Structure) that describes the resources needed and the functions provided by the card. Unfortunately, many PCMCIA card manufacturers have provided incomplete or erroneous CIS data, and for these cards it is not possible to perform fully automatic configuration based solely upon the information provided by the card. For this reason the server maintains an internal database of configuration descriptions based upon the card name and manufacturer. When a card is inserted the server will retrieve the card name and manufacturer and then search the database for a matching entry. This entry contains information that the server can use to find an appropriate configuration for the card. These configuration entries are stored in the file **/etc/config/pcmcia.cards**, new entries may be added to match any cards that will be used. The **pin** utility can create these entries automatically, but due to deviations from or misinterpretations of the PCMCIA standard by some manufacturers, they must often be manually edited.

The **pin** utility is invaluable for creating configuration entries for new I/O cards. The **pin config** command will attempt to create configuration entries for any cards currently installed. There are several situations that may require manual editing of the configuration file:

- The card does not have a valid function identification within the CIS. This will cause the generated configuration to have an invalid card type in the configuration lines. The type must be changed to reflect the actual card type.
- Some cards may require voltages other than 5V on the **V_{pp}** line. A **vpp** line will have to be added to the configuration entry.
- The configuration will have a **register = 0,0x40,0x40** line to set the card to level interrupts. Some cards do not operate correctly with level interrupts and this line must be removed or commented out.
- The order of the configurations is incorrect. The server will attempt each configuration in the order it appears in the file, a different order may be desired in order to control the resources used by a card.

It is also useful to use the **pin cis** command to check the generated configuration entry against the card CIS data.

Once an appropriate configuration has been found the server will set a configuration and allocate resources for the card. The server may also inform processes that a card has been inserted.

Card Configuration Database Entries

In the following description all numeric constants follow the same syntax: decimal numbers start with **1** to **9**, hexadecimal number start with **0x** and octal numbers start with **0**.

All card entries must all start with the line:

```
[device]
```

and must be followed by one or more of the following:

manufacturer = "*match-string***"**
product = "*match-string***"**
info1 = "*match-string***"**
info2 = "*match-string***"**

match-string is a string enclosed in double quotes. The special characters **?** and ***** match any character and any string of characters, respectively. Any other character will match only itself. For example:

```
abcde?gh*jk
```

will match **abcdefghjk** and **abcdexghcatjk** but not **abcdfxghcatjk** or **abcdeghcatjk**.

These lines are used to specify the card that this configuration applies to. Upon card insertion the server will compare these strings with information retrieved from the CIS. At least one of these lines must be included.

regbase = *addr*

addr is a numeric constant.

This line tells the server where in the card register space the configuration registers reside. If this line is not included the server will default to 0x100.

vcc = *voltage*
vpp = *voltage*

voltage is a numeric constant, and may have one digit after the decimal place. These lines tell the server what voltage the card expects on the $\mathbf{V_{cc}}$ and $\mathbf{V_{pp}}$ pins. The default value is 5.0 volts. Some modems require that $\mathbf{V_{pp}}$ be set to 12.0 volts for proper operation, most other cards will operate correctly with the default values. CAUTION: it is possible that some cards could be damaged by an improper setting of the **vpp** parameter.

register = *register*, *value*[, *mask*]

register is a card configuration register number. Register 0 is the configuration option register, register 1 is the configuration

and status register, register 2 is the pin replacement register, and register 3 is the socket and copy register.

value is a numeric constant and specifies the value to be written to the register.

mask is an optional numeric constant. If *mask* is not supplied *value* will be written directly to the register. If *mask* is specified then only bits that have a corresponding bit set in the mask will be altered by *value*. For instance if the register initially contains 0x35, *value* is 0x02, and *mask* is 0x12 the new register value will be 0x27.

A common use for this command is to set bit 6 of the configuration option register to place the card into level interrupt mode:

```
register = 0, 0x40, 0x40
```

For more information on the definitions of the various registers, refer to the document *PCMCIA PC Card Standard* from the PCMCIA (Personal Computer Memory Card International Association).

config = *index*, *type*, *resource*[, *resource*]...

index is a numeric constant that indicates the configuration index value to be written to the card configuration register. *type* is the generic card type that this configuration represents. The high order byte is the generic type and the low order byte is the specific subtype. The following values are defined:

Type	Definition
0x0201	Serial/modem devices with 8250/16450 UARTS
0x0301	PC parallel ports
0x0401	ATA hard disk

continued...

Type	Definition
0x0501	VGA adapter
0x0601	NE2000 compatible network card
0x0602	NCR Wavelan card
0x8001	Adaptec T340 SCSI adapter (NCR53C400)
0x8101	New Media .WAVjammer

Configuration *resource*s are used to specify what resources a configuration requires and can be specified in the form:

irq *irq*[**|***irq*]...

irq is a numeric constant that indicates a single IRQ level. Multiple IRQs can be specified by separating them with **|** characters. If this card will work with any IRQ, the string "**Any**" can be used. You can have only one IRQ resource per configuration line.

io *range* [**(***iooption* [**,***iooption*]...**)**]

range is a port range. It can take one of several different forms:

*start***-***end*
*start***+***length*
Any+*length*

where *start*, *end* and *length* are numeric constants.

iooption can be one of:

width=8
width=16
offset=*offset*

where **width=8** specifies an 8-bit port, **width=16** specifies a 16-bit port, and **offset=***offset* specifies the address that will be

presented to the card when the host makes an access to the given port range. This is useful for cards that can only access fixed port ranges, as it allows the server to place the card anywhere in the host I/O space. Some host adapter chips do not support this capability, and any configurations with the offset option will be ignored. (Currently no servers support this option.)

mem *range* [(*memoption* [**,***memoption*]...**)**]

range is a memory range, and is specified similarly to port ranges. All memory ranges must be true physical addresses that are aligned to 4K boundaries.

memoption can be one of:

offset=*offset*	*offset* is a numeric constant that specifies the base card address to which the memory window will map.
attribute	The default is to map common memory, but if it is desired that attribute memory be mapped, use the attribute option.

When a card with matching identification strings is inserted the server will try each configuration line until one succeeds.

Here is a sample entry for the Socket Communications EA+ Adapter (an NE2000 compatible Ethernet card):

```
[device]
manufacturer = "Socket Communications Inc"
product = "Socket EA PCMCIA LAN Adapter Revision D"
regbase = 0x10000
config = 0x03, 0x0601, irq any, io 0x2a0-0x2bf (width=16)
config = 0x05, 0x0601, irq any, io 0x2c0-0x2df (width=16)
config = 0x07, 0x0601, irq any, io 0x2e0-0x2ff (width=16)
config = 0x09, 0x0601, irq any, io 0x300-0x31f (width=16)
config = 0x0b, 0x0601, irq any, io 0x320-0x33f (width=16)
config = 0x0d, 0x0601, irq any, io 0x340-0x35f (width=16)
config = 0x0f, 0x0601, irq any, io 0x360-0x37f (width=16)
config = 0x01, 0x0601, irq any, io any+32 (width=16)
register = 0, 0x40, 0x40     ; level mode interrupts
```

© 2005, QNX Software Systems

Files:

/etc/config/pcmcia.cards

Card descriptions and configurations

/etc/config/pcmcia.cfg
/etc/config/pcmcia.*node*

Default resources

Exit status:

The **Pcmcia.*** servers terminate only upon encountering errors during initialization or upon getting a signal (i.e. being killed).

0 The server has shut down successfully.

>0 An error occurred during initialization.

See also:

PcmciaBeep, **PcmciaLaunch**, **pcmciatrap**, **pin**

Syntax:

Print command line that would be required to start the appropriate driver:

pcmciatrap [*options***] [query]**

Start driver:

pcmciatrap [*options***] start**

Options:

-q Quiet output.

-v Verbose output to help debugging failed detection.

Description:

The **pcmciatrap** utility will look for multiple PCMCIA adapters installed on the machine, including the ones on the PCI bus. If **start** is not specified, the command line that would be required to start the appropriate driver will be written to the standard output. If **start** is specified, the appropriate PCMCIA driver (**Pcmcia.***) will be automatically started. This is ideal for use in **sysinit** files.

WARNING: The pcmciatrap utility determines the chip type by probing I/O ports that are likely to be associated with a PCMCIA adapter. If you have other adapters at these addresses the utility may confuse the hardware and/or any associated drivers. Some NE2000 Ethernet adapters can crash the machine if they are accessed improperly.

The PCMCIA adapter hardware that **pcmciatrap** scans for includes the following:

Hardware	I/O Ports
Intel 386sl	0x3e0, 0x3e2
Cirrus 67xx	0x3e0, 0x3e2

Additional chips may have been added since this document was published. See the usage message for **pcmciatrap** for a list of currently supported PCMCIA chips. (**use pcmciatrap**)

The **pcmciatrap** utility may be run only by **root**.

Files:

The standard input is unused.

The **query** information and **-v** output is written to the standard output.

Exit status:

>0 Failed to detect PCMCIA adapter and/or start driver.

0 Successfully detected adapter/started driver.

See also:

Pcmcia.*, **PcmciaLaunch**, **pin**

Syntax:

pin [-n *node***] [-s** *socket***] [***command***]**

Options:

-n *node* Display information about the PCMCIA server running on the specified node.

-s *socket* Display information about this PCMCIA socket only (starting at 1)

command The type of information to display:

Command	Description
status	Show status of each socket (default)
resources	Show port, IRQ and memory resources
config	Attempt to make configuration file entries
cis	Decode CIS on card
clear	Clear all resource tables on server
file	Cause server to reread configuration files
translate *file*	Make configuration file entries from CIS on standard input or *file*
tuples *file*	Decode CIS from standard input or *file*

Description:

The **pin** utility displays useful information about PCMCIA resources, servers and cards, and can also be used to create configuration file

entries for PCMCIA cards (these entries must often be “tweaked” by hand — see the **Pcmcia.*** PCMCIA server documentation for details).

The QNX PCMCIA server will automatically configure PCMCIA I/O cards as they are inserted. It will also manage computer resources such as memory windows, ports and IRQs. Resources will be allocated as cards are inserted and then freed as cards are removed. Drivers that are PCMCIA-aware can be informed of card insertions and removals based upon the type of card. These drivers can then configure themselves to use the correct ports and IRQs. The **PcmciaLaunch** utility will automatically start drivers and pass them command-line arguments describing the ports and IRQs used by a card, thus permitting a non-PCMCIA-aware driver to be automatically started when a card is inserted and stopped when removed.

Each I/O card must contain a CIS (Card Information Structure) that describes the resources needed and the functions provided by the card. Unfortunately, many PCMCIA card manufacturers have provided incomplete or erroneous CIS data, and for these cards it is not possible to perform fully automatic configuration based solely upon the information provided by the card. For this reason the server maintains an internal database of configuration descriptions based upon the card name and manufacturer. When a card is inserted the server will retrieve the card name and manufacturer and then search the database for a matching entry. This entry contains information that the server can use to find an appropriate configuration for the card. These configuration entries are stored in the file **/etc/config/pcmcia.cards**, new entries may be added to match any cards that will be used. The **pin** utility can create these entries automatically, but they must often be manually edited due to deviations from or misinterpretations of the PCMCIA standard by some manufacturers.

Once an appropriate configuration has been found the server will set a configuration and allocate resources for the card. The server may also inform processes that a card has been inserted.

Creating I/O Configurations

The **pin** utility is invaluable for creating configuration entries for new I/O cards. The **pin config** command will attempt to create configuration entries for any cards currently installed. There are several situations that may require manual editing of the configuration file:

- The card does not have a valid function identification within the CIS. This will cause the generated configuration to have an invalid card type in the configuration lines. The type must be changed to reflect the actual card type.
- Some cards may require voltages other than 5V on the Vpp line. A **vpp** line will have to be added to the configuration entry.
- The configuration will have a **register = 0,0x40,0x40** line to set the card to level interrupts. Some cards do not operate correctly with level interrupts and this line must be removed or commented out.
- The order of the configurations is incorrect. The server will attempt each configuration in the order it appears in the file. A different order may be desired in order to control the resources used by a card.

Several of the **pin** commands can be used to assist in creating configuration entries:

- **pin cis** can be used to check the generated configuration entry against the card CIS data.
- **pin tuples** may be used to read a CIS from the standard input or a file
- **pin translate** will generate a configuration entry from a CIS from a file or standard input

© 2005, QNX Software Systems

Exit status:

>0	Failed
0	Success

See also:

Pcmcia.*, **PcmciaLaunch**

Syntax:

```
Pipe &
```

Options:

None.

Description:

Pipe implements the subset of file I/O known as pipes (or anonymous FIFOs). A simple form of interprocess communication, pipes are mostly used to connect the output of one process to the input of another to form a series of filters.

Although **Fsys** also implements pipes, **Pipe** doesn't need a disk to act as the pipe buffer, as it does all its buffering in memory. Hence you can use **Pipe** on diskless workstations. Also, since it's optimized for pipe I/O — that's all it does — **Pipe** often achieves better pipe throughput than **Fsys**. When a machine is running **Fsys** it is possible and normally desirable to also run **Pipe**. When this is done, all pipes will go through **Pipe**.

You must be **root** to run **Pipe**.

Examples:

Start the pipe manager:

```
Pipe &
```

Files:

The **Pipe** manager closes its standard input, standard output and standard error immediately after its startup initializations have been completed. Any errors that occur during startup will result in error messages being written to standard error.

© 2005, QNX Software Systems

Exit status:

Pipe terminates only if an error occurs during startup or if it has been started in the foreground and has forked a copy of itself as a background process.

0 **Pipe** has successfully forked a copy of itself in the background.

>0 An error occurred during initialization. A diagnostic message will have been written to standard error.

See also:

Fsys

Syntax:

pr [+*page***] [-***column***] [-a] [-d] [-e[***char***][***gap***]]**
[-f] [-h *header***] [-i[***char***][***gap***]] [-l** *lines***] [-m]**
[-n[*char***][***width***]] [-o***offset***] [-r] -s[***char***] [-t]**
[-w*width***] [-c | -v] [-b]**
[*file***...]**

Options:

The options for **pr** do not follow the standard utility syntax rules. The options **-l** ("el"), **-o**, **-s**, and **-w** don't allow the option letter to be separated from its argument. The options **-e**, **-i**, and **-n** require that both arguments, if present, not be separated from the option letter.

+*page*
Begin printing at the page number of the formatted input that is specified by *page*.

-*column*
Print output in columns that is *column* spaces wide and is in text column rank order (default number of columns is 1). When you specify this option, options **-e** and **-i** are assumed for multiple text-column output. If you specify the **-t** option, **pr** uses the minimum number of lines to display the output.

The **-m** option is not allowed when the **-***column* option is used.

-a
Print multiple-column output across the page. You should use this option only with the **-***column* option.

-b
(GNU extension) Balance columns on the last page.

-c
(GNU extension) Print unprintable chars as control prefixes (^*c*)

-d
Print double-spaced output; append an extra newline character after every newline found in the input.

-e [*char*] [*gap*]

Expand input tabs to column positions *gap*+1, 2×*gap*+1, 3×*gap*+1, etc. If *gap* is zero or is omitted, **pr** assumes default tab settings at every eighth position. All tab characters in the input are expanded into the appropriate number of space characters. If you specify any non-digit character, *char*, that character is treated as the tab expansion character.

-f (GNU extension) Synonym for **-F**, below.

-F Use a form-feed character for new pages (default is to use a sequence of newline characters). Prior to displaying the first page of output, an alert character (bell) is written to standard output.

-h *header* Use the string *header* for the header, instead of the filename.

-i [*char*] [*gap*]

In output, replace multiple space characters with tabs wherever two or more adjacent space characters reach column positions *gap*+1, 2×*gap*+1, 3×*gap*+1, etc. If *gap* is zero or is omitted, **pr** assumes default tab settings at every eighth position. If you specify *char* (any non-digit character), that character is used as the output tab character.

-l *lines* (“el”) Set the page length to *lines* (default is 66 lines). If *lines* is smaller than the number of lines required for the header and trailer, **pr** suppresses both the header and the trailer, as if the **-t** option were in effect.

-m Merge files. Standard output is formatted so that **pr** writes one line from each specified file side by side into text columns of equal fixed widths, in terms of the number of column positions.

-n [*char*][*width*]

In output, provide line numbering that is *width* digits wide (default width for line numbering is 5 digits). The number occupies the first *width*+1 column positions of each text column or each line of **-m** output. If *char* (any non-digit character) is given, that character is appended to the line number to separate it from whatever follows (default for *char* is a tab character). The maximum width you may specify is 9.

-o*offset* Precede each line of the output by the number of space characters specified by *offset*. (default number of spaces is zero). The number of column positions per line is the sum of the width (from the **-n** option) and the offset.

-r Don't print diagnostic reports on failure to open files.

-s[*char*] Separate text columns by the single character *char* (default for *char* is the tab character).

-t Print neither the identifying header nor the trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.

-v (GNU extension) Print unprintable chars as octal escapes (*nnn*)

-w*width* Set the width of the line to *width* column positions for multiple text-column output only (default is 71).

file The pathname of a file to be printed. If you don't specify any files, or if *file* is the dash character (**-**), the standard input is used.

© 2005, QNX Software Systems

Description:

The **pr** utility is a printing and pagination filter that prints files to the standard output.

By default, the listing is separated into pages, each with a header that includes the page number, date, time, and the file's pathname. The header is of the form:

\n
\n
MMM DD HH:*MM YYYY* **Page** *N***\n**
\n
\n

The trailer is simply the 5 newline characters required to advance to the next top of form (unless **-f** is used, in which case a form-feed will be sent). Text columns are of equal width; there is at least one blank separation character between text columns. Lines that don't fit into a text column are truncated. If standard output is associated with a terminal, diagnostic messages are suppressed until **pr** has completed processing.

Environment variables:

TZ Specifies the local time zone for use in writing header lines.

Signals:

If **pr** receives an interrupt while printing to a terminal, it flushes all accumulated error messages to the screen before terminating.

Exit status:

0 All files were written successfully.

>0 An error occurred.

Author:

GNU

Caveats:

Old (QNX 4.22 and earlier) versions of **pr** would fold long lines. Versions of **pr** shipped with QNX 4.23 and later do not. Use the **fold** utility to fold lines before submitting the file to **pr**.

See also:

fold, **lp**

prefix

© 2005, QNX Software Systems

Examine or change the pathname prefix tree (QNX)

Syntax:

prefix [-A *prefix***=***string***] [-D** *prefix***]**
[-n *node_id***] [-R** *prefix***=***string***]**
[-s]

Options:

-A *prefix***=***string*

Add prefix with alias *string*.

-D *prefix* Delete the specified prefix. If the prefix is directly owned by a resource manager (is not an alias prefix), unmount the device. (See **umount**.)

-n *node_id* Print the prefix string from the indicated node.

-R *prefix***=***string*

Replace the specified prefix with new alias *string*. This is the equivalent of deleting the prefix (**-D** *prefix*) then adding the prefix (**-A** *prefix***=***string*).

-s When adding/changing an alias prefix, skip checks for real mount points.

Description:

The pathname space is administered through the use of two forms of prefixes:

1 prefixes registered by I/O managers

2 aliases

The **prefix** utility allows you to examine and modify the pathname prefix tree.

The prefix tree is essentially an in-memory pathname redirector that is maintained on each computer. When a file is opened, its pathname is

matched against the prefix strings in order to direct the request to open the file to the appropriate I/O manager.

For example, the character device manager **Dev** registers the **/dev** prefix. Therefore, when a process calls *open()* with **/dev/xxx**, a prefix match of **/dev** will occur and the *open()* will be directed to **Dev** (the prefix's owner).

The form of an I/O manager prefix is as follows:

*prefix***=***pid***,***unit*

where *pid* is the process ID of the I/O resource manager and the *unit* number is a single character assigned by the I/O manager. For example, if **Fsys** is process 3 and **Dev** is process 5, then the prefix tree might appear as:

```
/=3,b
/dev=5,a
/dev/hd0=3,a
/dev/hd0t77=3,d
```

Note that **Fsys** has registered three prefixes:

- the root (**/**)
- a block special file representing the entire hard disk (**/dev/hd0**)
- a block special file representing a QNX 4 partition (**/dev/hd0t77**)

In cases where prefixes overlap, the longest match always wins. Therefore, in the example above, references to **/dev/hd0** and **/dev/hd0t77** are directed to **Fsys** rather than to **Dev**, while all other pathnames starting with **/dev** are still forwarded to **Dev**.

Aliases

In addition to this absolute redirection to an I/O manager, another form of prefix, called an *alias*, lets you replace a matched prefix with a new prefix, which is then re-applied to the prefix tree. Unlike the

© 2005, QNX Software Systems

I/O manager form of prefix, aliases can be modified using the **prefix** utility. An alias is of the form:

*prefix***=***replacement_string*

Aliases are often used to map remote resources into the local pathname space. For example, a diskless workstation might typically create an alias for its root (**/**) to a filesystem on another node:

```
prefix -A /=//10/
```

Aliases can also be used to create special device names. For example, the following command:

```
prefix -A /dev/console=//20/dev/con1
```

would alias **/dev/console** to the primary console on node 20.

The **-A** option adds a new prefix to the prefix tree. The prefix being added must not already be in the tree.

The **-R** option deletes, then adds, a prefix to the prefix tree.

The **-D** option deletes an existing prefix from the prefix tree.

Examples:

List the prefix tree currently in effect:

```
prefix
```

List the prefix tree in effect on node 20:

```
prefix -n 20
```

Create prefix aliases for **/home1**=**//10/home1** and **/home2**=**//14/home2**:

```
prefix -A /home1=//10/home1 -A /home2=//14/home2
```

Create a prefix alias for **/dev/printer** pointing to **//12/dev/spool/text**:

```
prefix -A /dev/printer=//12/dev/spool/text
```

Remove the prefix for **/dev/printer**:

```
prefix -D /dev/printer
```

Caveats:

Since redefining the prefix aliases affects all the processes on the computer, you should do this with caution, especially when dealing with definitions for root (**/**).

See also:

ln

System Architecture

printf

© 2005, QNX Software Systems

Write formatted output (POSIX)

Syntax:

printf *format* **[***argument***...]**

Options:

format A string describing how the named *argument*s are to be formatted.

argument A string to be written to standard output, under control of *format*.

Description:

The **printf** utility writes formatted arguments to standard output under control of the *format* control string (the syntax for *format* is based upon the *printf()* function).

The *format* control string contains the following types of characters:

- ordinary characters that are written exactly as they occur in the *format* string
- escape sequences that represent nongraphic characters
- conversion specifications that specify the output format of the named *argument*s

You introduce a conversion specification by the **%** character. To literally display this character, you encode it as **%%**. Immediately following the **%** character, you may specify any of the following, in order:

<*flags***><***width***><***precision***><***type len***><***conversion***>**

where:

flags are the format control flags

width is the field width

precision	is the numerical precision
type len	is a type length modifier
conversion	is the conversion character

These components are processed left to right.

Format control flags

The format control flags modify the meaning of the conversion specification. You can specify zero or more of these flags, in any order:

-	Left justify output within the field.
+	Always begin the result of a signed conversion with a sign (**+** or **-**).
Space	If the first character of a conversion is *not* a sign, prefix a **space** to the result.
#	Convert the value to an alternate form:

For this conversion:	The result is formatted as follows:
o (octal)	the first digit is zero
X or **x** (hex)	a nonzero result has **0X** or **0x** prefixed to it
e, **E**, **f**, **g**, and **G**	the result always has a decimal point (**.**), even if no digits follow
g and **G**	trailing zeros aren't removed from the result as they usually are
c, **d**, **i**, **s**, and **u**	no effect; the **#** is ignored

continued...

© 2005, QNX Software Systems

For this conversion:	The result is formatted as follows:
0 (zero)	For all conversions, pad the field width with leading zeros. The **0** flag is ignored for the **d**, **i**, **o**, **u**, **X**, and **x** conversions if a precision is specified; it's also ignored if the **-** flag is given.

Field width

An optional decimal integer that specifies the minimum number of characters for displaying the formatted item. If no field width is specified, or if the value is less than the number of characters required to represent the converted value (subject to the specified precision), a field of sufficient width to contain the converted value is used.

If the number of characters required to represent the converted value is less than the specified field width, the value is padded on the left (or on the right if the **-** format control flag is given) with spaces or "**0**" (zero) characters. If the field width begins with a zero, the value is padded with zeros, otherwise spaces.

Precision

An optional decimal integer following a period character (**.**). The effect of the given precision depends on the conversion character you specify (see "Conversion character," below).

Type length

An optional character that modifies the interpreted size of the conversion character. The only supported type length character is **l** ("el"), which causes a **d**, **i**, **o**, **u**, **X**, or **x** conversion to process a **long int** or **unsigned long int** argument.

Conversion character

A character that determines the type of conversion to be applied. For example, if **d** is specified, **printf** treats the corresponding argument as a decimal integer. The conversion characters are:

d i o u X x

The argument is an integral value. It is printed as one of the following:

signed decimal (**d** or **i**)
unsigned octal (**o**)
unsigned decimal (**u**)
unsigned hexadecimal (**X** or **x**)

For each of these, the specified precision is interpreted as the minimum number of digits to appear (defaults to 1 if no precision is specified). In the absence of a type length specifier, these conversions are either signed in the range -32768 to 32767 or unsigned in the range 0 to 65535. If a type length is specified, the conversions are either signed in the range -2147483648 to 2147483647 or unsigned in the range 0 to 4294967295. For the hex format, the case of the specifier (**X** or **x**) implies the case of the output.

e, **E** The argument is a floating-point value, and is displayed in scientific notation. The value is printed in the style:

[**-**]*d*[**.***ddd*]**e{**+**|-}***dd*

The first part is an optional sign. The second is a single digit, which is nonzero if the argument is nonzero. The third (optional) part — a decimal character followed by a number of digits equal to the specified precision — is printed if the precision is nonzero. If the precision isn't specified, it's assumed to be 6. If the precision is 0, the decimal isn't printed. The fourth part is an **e** or **E**, based upon the case of the conversion character, followed by a signed value. This value represents the order of magnitude.

f The argument is a floating-point value. The value is printed in the style:

[-]*ddd.ddd*

where the number of digits printed after the decimal character is equal to the specified precision. If the precision isn't specified, it's assumed to be 6. If the precision is 0, the decimal isn't printed.

g, **G** The argument is a floating-point value. The value is printed in the "**e**" style (or "**E**" style if a **G** conversion character is given) if the exponent in this style would be less than -4 or greater than or equal to the specified precision; otherwise, the value is printed in the "**f**" style.

c The first character of the argument is printed.

s The argument is a string of characters and the string is printed. The number of characters printed from the string is equal to the precision, if specified. The string is left- or right-justified within the specified field width. If no field width is specified, no padding is applied.

Escape sequences

The **printf** utility interprets the character escape sequences within the *format* string in a way similar to the C programming language. These sequences, which you introduce via the backslash character (\), are translated as follows:

This sequence:	Writes:
\a	an **alert** character (bell)
\b	a **backspace** character
\f	a **form-feed** character

continued...

This sequence:	Writes:
\n	a **newline** character
\r	a **carriage return** character
\t	a **tab** character
\v	a **vertical tab** character
\'	a single quote (') character
****	a backslash (\) character
*****d*	the character specified by the one-, two-, or three-digit octal number *d*

Examples:

Display the string "**hello, world**" on a line by itself:

```
printf "hello, world\n"
```

Do the same as above, but don't output a **newline**:

```
printf "hello, world"
```

Display a value in scientific notation:

```
printf "n = %e\n" 3.1415926535897
```

Exit status:

0 Successful completion.

>0 An error occurred.

© 2005, QNX Software Systems

See also:

echo, **print**

Syntax:

Proc32 [*options***]...**

Options:

The options for **Proc32** do not follow the standard utility syntax rules. Options and option arguments must all be supplied as separate command-line arguments (i.e. separated by white space on the command line). Options may not be strung together and must be separated from their arguments by white space.

-a Disable accounting.

-b[**1|2|3**] Specifies reboot method:

- *default* Use system port 0x92, then keyboard controller, then cpu shutdown.
- **-b** Do not use keyboard controller when executing the default sequence.
- **-b1** Reboot via keyboard controller first, then execute the default sequence.
- **-b2** Reboot via system port 0x92 first (which is the default sequence).
- **-b3** Reboot via processor shutdown only.

The options **-b1**, **-b2** and **-b3** may be combined with **-b** which subtracts the keyboard method from the subsequent execution of the default reboot sequence as follows:

Option	Sequence
default	port 0x92, keyboard, cpu

continued...

Option	Sequence
-b	port 0x92, cpu
-b1	keyboard, port 0x92, keyboard, cpu
-b1 -b	keyboard, port 0x92, cpu
-b2	port 0x92, keyboard, cpu
-b2 -b	port 0x92, cpu
-b3	cpu
-b3 -b	cpu

Interrupt vector 0xf5 is always invoked before trying any of these methods. For custom hardware, you can point this vector at code to cause the restart.

-B *node* Set base node in network (default is 1)

-d *num_major* The number of major device numbers to be supported (default is 64).

-D If the debugger (**Debugger32**) is bound into the operating system, suppress its automatic invocation upon boot.

-e *num_semaphores*

Specify the maximum number of semaphores available in the system (default is 128).

-f *min max total* This option controls the availability of file descriptors (FDs):

min The minimum number of FDs that must be available before a process will start. This argument defaults to 16 and should be left at that value if POSIX-compliance is a requirement (POSIX requires that a

process be capable of opening at least 16 files). You can save some memory by reducing this number.

max The maximum number of FDs that can be available for a process (default is 512).

total The total number of FDs available in the system (default is *num_procs*×16).

Each process starts off by taking *min* FDs, which it may grow to *max* FDs.

-F *timer_freq* Set frequency of the 8254 clock (default is 1193181 Hertz).

-H Allocate heap as needed.

-i *irq* Make *irq* the highest-priority hardware interrupt in the system. You can specify a number from 0 through 7 (default is 3).

-l *nid*
-l @*bios_offset*
-l @**w***paddr*
-l **!**[**b**|**w**|**l**]*ioport*

("el") Assign a logical node ID to this machine (default is 1). Each connected machine must have a unique logical node ID. A node ID may be specified directly as a decimal integer (*nid*) or read from the BIOS data area (@*bios_offset*), from the physical address (**@w***paddr*), or from an I/O port (**!***ioport*). The forms of the node specification are as follows:

nid Specifies the node ID directly (decimal).

@*bios_offset* Extract the node ID from the (byte) memory area at offset *bios_offset* (hex) in the BIOS data area.

@w*paddr*	Extract the 16-bit node ID from the physical address *paddr* (hex).
!*ioport*	Extract the node ID by reading a byte from the I/O port specified by *ioport* (hex).
!b*ioport*	Extract the node ID by reading a byte from the byte-wide I/O port specified by *ioport* (hex).
!w*ioport*	Extract the node ID by reading a word from the word-wide I/O port specified by *ioport* (hex).
!l*ioport*	Extract the node ID by reading a longword from the longword-wide I/O port specified by *ioport* (hex).

-L[**1**|**2**] Local. Don't permit outbound network access from this computer.

default	Allow inbound and outbound network access.
-L	Local. Restrict outbound network access.
-L1	Local. Restrict outbound network access for all users except **root**.
-L2	Disallow *inbound* network access, but allow unrestricted outbound access.

-m *num_shared_segs*

The number of shared segments (managed by the *qnx_segment_*()* family of functions) that will be available in this operating system image (default is 2000).

-M *machine_name*

The symbolic name for this computer (default is AT or PS/2). Some programs check for this name

and alter their behavior accordingly. For example, the floppy disk driver (**Fsys.floppy**) and some PS/2 disk drivers look for the name PS/2.

-n *num_names* The maximum number of names that can be registered locally by the *qnx_name_attach()* function (default is 80).

-o *port*[**,***baud*] Emit messages to the specified serial port (specified in hexadecimal). The baud may also be specified (decimal); the default baud is 9600.

-p *num_procs*[**,***code*]

The total number of real processes, virtual circuits, and proxies that can exist at any time (default is 500; maximum is 2000). The optional "**,***code*" is for command-line compatibility with earlier **Proc**s and is ignored.

-P *priority* Set **Proc**'s priority to the specified value between 26 and 30. (Default: 30)

-r *prefix_len* The size of the system prefix space (default is 2000).

-s *num_sessions* The maximum number of sessions that will be supported (default is 64).

-S *gdts ldts* Set the maximum number of global segments and local segments in the system (by default an appropriate value will be chosen). Most programs need two local segments, while few may need 30 or more. Global segments may be used by some programs for permanent shared memory.

-t *num_timers* The number of timers that will be supported (default is 100). These timers are allocated to processes with the *timer_create()* function and returned with the *timer_delete()* function. See also the *sleep()* and *alarm()* functions.

-T *trace_size* The size of the trace buffer, in kilobytes (default is 4K). See also the **tracectrl**, **traceinfo**, and **tracelogger** utilities.

-v *level* Set the verbosity *level* (0 to 3; default is 0). The message levels are as follows:

- Level 0: An internal inconsistency was detected. Contact QNX Software Systems.
- Level 1: The Kernel was forced to take a planned but uncommon recovery action.
- Level 2: Expected actions have occurred as a result of a misbehaving process or remote computer.
- Level 3: Status messages during normal operation.

-W Enforce page protection in kernel. Note that this could cause problems with debuggers or with older 32-bit graphics drivers.

-z *flag* Set the following flags:

- **N** Use traditional (i.e DOS) NDP faults (the default).
- **n** Use real NDP faults. Note that the **n** flag precludes the use of **int10**.
- **U** Prefer QNX behavior where discrepancies exist between Unix and QNX (the default).
- **u** Prefer Unix behavior where discrepancies exist between Unix and QNX.

Description:

Proc is a *required* process — it cannot be started at the command line, but must be built into the OS image.

The Process Manager (**Proc**) works closely with the kernel to provide essential operating system services. In fact, the kernel is bundled with the code for **Proc** and shares the same address space. For these reasons, **Proc** is a *required* system process in all operating system images that are built. You cannot start **Proc** after booting.

Most of the services **Proc** provides require the use of memory within **Proc**. The amount of memory used is controlled by tables whose sizes are set by options to **Proc**. Some tables are fixed in size, while other dynamically request more memory as needed.

Heapp (deprecated)
: displays the heap space for **Proc**

Heapf (deprecated)
: displays the heap space for FDs (file descriptors) (**-f** option).

Heapl
: displays the heap space for LDTs (local descriptor tables) (**-S** option).

Both *Heapp* and *Heapf* are now handled dynamically and are no longer needed.

In QNX documentation, you'll see the Process Manager referred to as **Proc**. However, in your buildfile you must specify **Proc32**. Note that **Proc32** allows both 16-bit and 32-bit programs to run.

See also:

/boot/sys/boot, **buildqnx**, **Debugger32**, **prefix**, **sin**, **sinit**, **tracectrl**, **traceinfo**, **tracelogger**

ps

© 2005, QNX Software Systems

Report process status (POSIX)

Syntax:

ps [*options***]...**

Options:

-A	Print information for all accessible processes.
-a	Print information for all processes associated with terminals except for process group leaders.
-d	(QNX extension) Print information for all processes except process group leaders.
-e	(QNX extension) Print in long format showing elapsed time (equivalent to **-F "%p %r %N %S %b %T %a"**).
-F *format*	(QNX extension) Print information according to the format specifications given in *format*.
-G *grouplist*	Not supported in QNX 4.24. See **-g**.
[**-g** *gid*[**,***gid*]...	(QNX extension) Print information only for processes whose group IDs are listed.
-l	("el" — QNX extension) Print in long format (equivalent to **-F "%p %r %N %S %b %x %a"**).
-n *node*	(QNX extension) Print information only for processes on the specified node.
-o *format*	Not supported in QNX 4.24. See **-F**.
-p *pid*[**,***pid*]...	Print information only for processes whose process IDs are listed.
-s *sid*[**,***sid*]...	(QNX extension) Print information only for processes whose session IDs are listed.

-S *state*[**,***state*]...
: (QNX extension) Print information only for processes whose states are listed. (e.g. READY, WAIT).

-t *sid*[**,***sid*]...
: Print information only for processes associated with the terminals listed. Terminals must be in the form **[***nid***]***tty_basename*, where *nid* is the node the device exists on, and *tty_basename* is the basename portion of the controlling terminal's pathname. (e.g. **[3]con1** for **//3/dev/con1**)

-U *userlist*
: Not supported in QNX 4.24. See **-u**.

-u *uid*[**,***uid*]...
: (QNX extension) Print information only for processes whose user IDs are listed.

Description:

The **ps** utility prints information about processes, subject to having the appropriate privilege to obtain information about those processes. With no options, **ps** prints information about processes associated with the current terminal. The output includes the process ID, terminal name, cumulative execution time, and command name of each process.

The options that use lists (**-g**, **-p**, **-t**, **-u**) can list multiple items separated by commas or white space, as long as all the items are contained within a single command-line argument. If white space is used, this will require quoting the list when invoking **ps** from the shell.

The initial set of processes selected by **-a**, **-A**, or **-d** is intersected with the processes selected by the **-g**, **-p**, **-t**, or **-u** options, if any of the latter are specified.

If none of **-a**, **-A**, or **-d** is specified, **ps** behaves as though you specified **-u** *your_uid*.

Controlling output

To control the content and format of output, you use **-F** *format* option, which takes any of the field descriptors listed below. All of these field descriptors, with the exception of **%%** and **%a**, are of the form:

%[**+** **-**][**"***title***"**]*x*

The **+** forces right justification; the **-** forces left justification. If neither **+** nor **-** is specified, the default justification for the descriptor is used. If no *title* is given for a field, **space**s are printed. The [and] aren't part of the descriptor; they simply show the optional parts of the descriptor.

The **%a** field descriptor has the form:

%[**+** **-**][*width*][**"<***TITLE***>"**]**a**

You can use the optional *width* for limiting the length of what is printed for this field.

Any text included between the field descriptors is printed on all lines, including the header line.

Field descriptors:

%% A literal **%** character.

%a The command with all its arguments as a string.

%b The process that this process is blocked on.

%c The name of the command (*argv[0]* value) as a string.

%C The ratio of CPU time used (**%x**) to CPU time available, expressed as a percentage.

%d The current working directory of the process.

%e The initial environment of the process.

%f The flags of the process.

%g The real group ID of the process.

%G The effective group ID of the process.

%p The decimal value of the process ID.

%P The decimal value of the parent process ID.

%n The decimal value of the process's nice value.

%N The decimal value of the process's priority.

%r The decimal value of the process group ID.

%s The decimal value of the session ID.

%S The process state of the process.

%t In the POSIX locale, the elapsed wall clock time of the process in the form:

[*dd*-]*HH*:*MM*:*SS*

The *dd* field is not shown if less than one day.

%T In the POSIX locale, the relative start time of the process in the form:

[*dd*-]*HH*:*MM*:*SS*

The *dd* field gives the number of days before the current date that have elapsed since the process started (not shown if *dd* is less than one day).

%u The real user ID of the process. If the user ID is present in the **/etc/passwd** file, a textual user name is printed; otherwise, a decimal numeric user ID is printed.

%U The effective user ID of the process (see **%u**).

%x In the POSIX locale, the cumulative CPU time of the process in the form:

[*dd*-]*HH*:*MM*:*SS*

The *dd* field is not shown if less than one day.

%y If a controlling terminal is present, the name of the terminal, consisting of the node ID followed by the basename of the device pathname.

%z The size of the process in (virtual) memory in kilobytes, expressed as a decimal integer. Shared code or segments are not accounted for.

Examples:

Emulate the default output from a System V **ps** command:

```
ps -F '%+"PID"p %+"TTY"y %+"TIME"t %+"COMMAND"C'
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

sin

Syntax:

pwd

Options:

None.

Description:

The **pwd** utility writes the pathname of the current working directory to the standard output.

The **pwd** command is available both as a shell alias (equivalent to **print "**$*PWD***"**), and as a standalone utility.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

cd, **fullpath**, **ls**, **sh**

qcp

© 2005, QNX Software Systems

QNX communications protocol (QNX)

Syntax:

Send files:

qcp [*device***] se [***options***]** *src_file***[,***dst_file***]...**
[x=*index_file***]...**

Receive files:

qcp [*device***] re [***options***] [-f** *filename***|-p** *prefix***]**

Options:

device	Pathname of serial device to use. The default is the device connected to *stdin* and *stdout*.
-f *filename*	Force received files to have this name.
-F	Required to receive files on a flash filesystem which does not support all the POSIX file control mechanisms.
-l *logfile*	("el") Append the file transfer event to the logfile.
-m	Suppress making directories for received files.
-n	Receive only files that are newer than existing files.
-p *prefix*	Place this path prefix on the names of any received files.
-q	Be quiet; display nothing during the transfer.
-r	Relax timing; double timeouts and quadruple retry counts.
-s *packet_size*	The size of transmitted data bursts (default is 2048 bytes).
-t	Apply today's date to received files.

-u	Unlink files that lack write permission. This allows new versions of the affected files to be received and written to disk.
-V	Be verbose; display error status while transferring files.
x=*index_file*	A list of filenames; **qcp** will send each file named in the list. Multiple **x=***index_file* options can be specified and intermixed with singly named files to transmit.

Description:

The **qcp** utility provides an error-checked file transfer protocol that **qtalk** uses to transmit or receive files. This protocol is both highly efficient on packet switched networks and highly reliable through the use of 16-bit CRCs (Cyclic Redundancy Check).

The **qcp** utility automatically sends files with their pathnames, attributes, permissions, and date fields intact. The **qcp** protocol is ideally suited for use over public packet switch networks (X.25), as well as direct modem-to-modem connections. If communication errors are encountered, portions of the file will automatically be re-sent until the far end acknowledges correct reception of the file.

If you're using **qtalk** to communicate with a remote system, you can both send and receive files to the remote system with the **qcp** utility. To cause a file to be sent from the remote system to your local system, type a command of the following form into the **qtalk** session you have connected to the remote system:

```
qcp se file1  file2,file3 x=file4
```

This will cause the remote end to send the files to you; **qtalk** will automatically start **qcp** to receive the file. The files sent will be:

- **file1**
- **file2** (**file2** is received with the filename **file3**)

- all the files named in the index file **file4**

You can create index files easily with the **ls -p** command.

To send a file to the remote system, type the following command into the remote shell:

```
qcp re
```

then use the Ctrl – A Ctrl – S key sequence to cause **qtalk** to send a file to the remote **qcp** utility.

If you want **qcp** to send a file through an explicitly named device, use a command of the form:

```
qcp /dev/ser1 se filename
```

where **/dev/ser1** is the port to send through. To receive from a particular device, use a command similar to:

```
qcp  /dev/ser1  re
```

You can abort a **qcp** file transfer in progress by pressing either Esc or spacebar. In turn, **qcp** displays a prompt asking for confirmation of the action. To confirm aborting **qcp**, you enter **y**. If a remote **qcp** in a receive state must be shut down, the following control-character sequence aborts it:

```
^V^X^X
```

Note that **qcp** will automatically remove any partially transferred files.

When transferring files via high speed modems, the **-s 16000** option is recommended.

See also:

ls, **qtalk**

qed

© 2005, QNX Software Systems

QNX fullscreen Editor (QNX2)

Syntax:

qed [-br] [*file* **[***commands***...]]**

Options:

-b	Browse only — do not allow the user to write the file.
-r	Restricted. Do not give the user the ability to start a shell or run other commands from the editor.
file	The name of the file to edit. There is no default; if the editor is started without a *file*, one will have to be specified within the editor before the text may be saved.
commands	A list of editor commands to execute on startup.

Description:

The **qed** utility is a fullscreen editor which has been shipped with QNX since the early days of the QNX2 operating system. It has been ported to QNX 4 for the convenience of the many people who have become accustomed to it. However, it is no longer the preferred editor in QNX and its use is deprecated. The **qed** editor is a niche editor available only on QNX, while other options such as **vi** and **emacs** are more universally available.

If you enter this editor by accident, the best way to get out is to hit **q** Enter while the cursor is on the top command line. Because it was designed for the console with full PC keyboard capabilities, its use is more limited on terminals and many terminfo entries do not describe the capabilities it requires for such basic things as getting back to its command line.

For more information on **qed**, you'll find a complete HTML manual online.

This editor is provided “as is” and its use is not encouraged.

Examples:

Edit the file **myfile.c**:

```
qed myfile.c
```

Edit the file **myfile.c**, moving to line 320 on startup:

```
qed myfile.c 320
```

Files:

/usr/lib/ed.macros

The **qed** utility uses internal macros for most of its operations. These macros are defined in this file.

/usr/lib/box.macros

This file contains a set of macros for box-drawing within **qed** using the PC character set line-drawing characters. This file is optional and may not be found on your system.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

vi, **vedit**

qtalk

© 2005, QNX Software Systems

Talk over communications line (QNX)

Syntax:

qtalk [*options***] [***system***]**

Options:

-b [*baud*|*data*|*parity*|*stop*][**,**...]

Change serial port to this baud, parity, stop bits, and/or data bits. Values of 1-2 are interpreted as stop bits, 7-8 as data bits, **none**, **even**, **odd**, **mark** and **space** as parity, all other numbers as baud. Order is not significant e.g. **-b 9600,8,n,1**

-c *hh*

Set this as the character (hexadecimal; no leading **0x**) that invokes **qtalk** commands. Default is **01** (Ctrl – A).

-d *hh*

Replace ASCII rubout with this character (hexadecimal; no leading **0x**). Default is **7f**.

-D *delay*

Wait this many 1/20th sec periods before running the command specified by the **-x** option.

The qtalk utility reads any data emitted by the modem during this delay period and displays the data on your screen before the command is started.

If you use **-x** without this option, **qtalk** won't wait to start the command. As a result, if the modem emits any information once **qtalk** has emitted the dialing string, you have no way of predicting whether **qtalk** or the command will see that information.

-e

Enable local echo.

-E

(Deprecated) Suppress **FF** codes (emulates **qnxs** terminals used in QNX 2). Equivalent to **-T qnxs**.

-h

Hang up the current modem line if someone dials a new system from within **qtalk** via the Ctrl – A command. By default, the current modem device is closed but the line isn't dropped.

-l *logfile* ("el") Log the **qtalk** session in *logfile*.

-m *modem*[**,***init_string*]

The name of the device to use, or the name of a modem pool to choose a device from. If you specify multiple **-m** options, **qtalk** will try them, in order, until it finds a device that's not in use.

If you specify an *init_string*, it will be emitted to the modem before anything else.

If the modem is defined in the modem pool file **/etc/config/qtalk.modems**, the *init_string* may be overridden by an *init_string* defined in the entry for the selected modem. The *init_string* defined closest to the actual naming of the selected device always supersedes other *init_string*s (for more information, see the discussion on modem pools).

-o *protocol***=***command*

Redefine transfer-protocol options, where *protocol* is one of the following:

This protocol:	Will do a:
qcp_se	qcp send
qcp_re	qcp receive
zmodem_se	ZMODEM send
zmodem_re	ZMODEM receive
other_se	*other send*
other_re	*other receive*

and where *command* is the command that will perform the file transfer. This command will be run by the Shell. The macro **$MODEM** will be set to the pathname of the modem device and, in the case of a send, the macro

$FILENAME will be set to the filename to be transmitted.

You can disable **qtalk**'s automatic invocation of a protocol by setting *command* to a null string (**""**). For example, the following would disable automatic ZMODEM receive:

```
qtalk -o zmodem_re=""
```

For information on default protocol commands or for more information on automatic invocation of protocols, see the section entitled "Invoking qcp and ZMODEM automatically" in the body of the utility description.

The other protocol lets you configure **qtalk** to use your own commands for sending and receiving files.

-P Ignore the 8th bit of received characters.

-s *system_directory*

Instead of **$HOME/.qtalk**, use this file to look up systems to dial.

-t *xfer_protocol*

Set the current transfer protocol to **qcp**, **zmodem**, or **other**. (To change the command string that will be run to perform the file transfer, use **-o**.)

-T *terminal_type*

Use this terminal emulation for screen output. The terminal type must be one of the following:

- **ansi**
- **pcansi**
- **qnx**
- **qnxs**

If you don't specify the **-T** option, **qtalk** will leave your console protocol as is.

See also the description of the **T** command in the Commands section.

-x "*command*"

Run this command after emitting the dialing string for the named system (see also **-D**). When the command is run, the **MODEM** environment variable is set to the pathname of the selected modem device.

system The name of the system you want **qtalk** to call. The system must be defined either in **$HOME/.qtalk** (or the file named by **-s**) or in the system-wide dialing directory, **/etc/config/qtalk**. For more information, see the description of the **d** (dial system) command.

Description:

The **qtalk** utility lets QNX users communicate with other computers via a serial line that's usually connected to a modem. The destination may be another host computer, in which case **qtalk** lets you use your computer as a terminal. The **qtalk** utility also lets two QNX users communicate and transfer files.

The **qtalk** utility sends any characters you type on the keyboard to the other system via the serial port. Any characters received through the port are displayed. In local echo mode, typed characters are echoed on the display as well as being sent out the serial port.

Configuring default behavior

The **qtalk** utility lets you to define new defaults for anything that can be specified by command-line options. You define these settings by creating or modifying a system (in the global and/or your personal dialing directory) called **defaults**. The first thing **qtalk** does is to look up the **defaults** dialing entry and process it, before looking at the command-line options that have been supplied. So, the **qtalk**

utility will look for **defaults** first in **$HOME/.qtalk** and, if not found there, then in **/etc/config/qtalk**. This allows new system-wide defaults to be set by a system administrator, while leaving individual users free to create their own default behavior for **qtalk**.

Command-line options defined in the **defaults** system are applied *before* the options specified in the actual command line. Thus you can override the settings in your defaults system with command-line options.

The **-s** command-line option doesn't change how **qtalk** searches for the **defaults** system.

Logging a session

You can log a **qtalk** session with the **-l** option:

```
qtalk -l /dev/par
```

or, if you wanted the log to go to the file **/tmp/logfile**:

```
qtalk -l /tmp/logfile
```

Using the command character

You can use a special command character to set special modes and options while you're within the **qtalk** environment. This special character defaults to Ctrl – A (^A) unless you change it with the **-c** option when you invoke **qtalk**. When you enter the command character, you will be shown some current settings and will be prompted to enter a command. If you enter the command character twice, a single command character is echoed to the modem. This allows you to send the command character to the remote system if you need to.

Replacing the rubout/delete character

You'll find the **-d** *hh* option useful when communicating with computers that have a rubout character different from the one you're used to. Many systems use the Backspace key (08 hex) to erase a character. QNX systems default to the ASCII rubout character (7F hex). If you type:

```
qtalk -d 08
```

qtalk will translate your Rubout key into backspace automatically.

Enabling flow control

Using very high-speed modems, or producing a log on slow printers or floppy disks, may cause some characters to be lost. To prevent characters from being lost in these cases, you can enable input flow control prior to invoking **qtalk** (see **stty**). This works only if the machine sending the data supports flow control of its output.

Transferring files

The **qtalk** utility lets you transfer files through either of two methods. You invoke the simpler method with the **w** (write) and **l** (log) commands to transfer text files to or from another system. No error checking is performed, however, so you should use this method only when communication lines are good (i.e. direct connect lines or reliable modem connections), or when the other system does not support any of the file transfer methods available from **qtalk**.

To send a file to a host using **w**rite, you should first make the host capable of receiving a stream of text. You could do this with an editor or the **cat** utility. You can then use the **w**rite command to send the file.

To receive a file from a remote host using the **l** (log) command, first turn on logging to the desired file (with the **l** command), then enter a command into the remote system to make it display the file (e.g. **cat** *filename* on a UNIX type system.)

The second method, involves the **s** (send) and **r** (receive) commands, providing access to more sophisticated protocols that include error checking and retransmission when transferring files to other systems.

Invoking qcp and ZMODEM automatically

The **qtalk** utility attempts to detect **qcp** and **zmodem** file transfers when **qtalk** is on the receiving end. If it detects one of these, **qtalk** automatically invokes the appropriate receive command for the protocols being used. You can disable this behavior; see the **-o** option.

When a **qcp** or **zmodem** transfer is detected, or when the **r**eceive command is given, **qtalk** will run (through a shell) one of the *protocol***_re** commands.

The corresponding *protocol***_se** commands (to send a file) are invoked only when the **s**end command is given; never automatically.

Here are the default protocol commands:

```
qcp_re="qcp $MODEM re"
qcp_se="qcp $MODEM se $FILENAME"
zmodem_re="rz <$MODEM >$MODEM"
zmodem_se="sz $FILENAME <$MODEM >$MODEM"
other_re=""
other_se=""
```

You can't automatically enable the *other_re* protocol. To invoke *other_re* or *other_se*, choose **s**end or **r**eceive from the command menu when the current file transfer protocol has been set to other.

Modem pools

The **qtalk** utility provides a modem pool facility that serves as a shorthand for specifying multiple modems (otherwise done by specifying multiple **-m** options). For example, if a modem pool named *hst* was set up which included the modems **//1/dev/ser1**, **//2/dev/ser1**, **//2/dev/ser2** and **//30/dev/ser2**, then instead of having to type:

```
qtalk -m //1/dev/ser1 -m //2/dev/ser1 -m //2/dev/ser2 -m//30/dev/ser2
```

you could simply type:

```
qtalk -m hst
```

The **qtalk** utility tries each device in each pool until it finds a device that's unused (i.e. no other process has the device open). When it finds an unused device, **qtalk** will do the following, in order:

1. emit an *init_string* (if there is one) to the modem
2. emit a dial string (if there is one)
3. run the command specified by **-x** (if **-x** was specified)
4. put you in an interactive session with the device

How to define a modem pool

You define modem pools in the **/etc/config/qtalk.modems** file. Here's the format of this file:

poolname *modem***[,***init_string***]...**
[<*whitespace***>** *modem***[,***init_string***]]...**
⋮

where:

poolname — An arbitrary name that identifies a group of modem definitions.

modem — Either the pathname of a modem device (e.g. **//3/dev/ser2**) or the name of another modem pool (e.g. *hst*). A *modem* definition can also include an optional initialization string (*init_string*).

If *modem* is a modem device (e.g. **//3/dev/ser2**) and the entry for that device contains an *init_string*, **qtalk** will, upon selecting that device, emit the string to the device and then emit any other communication to the device.

If *modem* is a modem pool (e.g. *hst*), **qtalk** will use the first available modem device in that pool. If the selected modem entry contains an *init_string*, that string will be emitted. If the selected entry

doesn't contain an *init_string*, but the line that references the modem pool does contain an *init_string*, that string will be emitted.

You can specify multiple modem definitions, separated by whitespace. You may continue onto the next line by starting it with one or more **space** or **tab** characters. If the next line doesn't begin with whitespace, it will be regarded as a new pool definition, not a continuation of the previous one.

If you specify an *init_string*, you can't insert whitespace between the modem, the comma, and the *init_string*. Also, if the *init_string* contains any **space**s or **tab**s, you must enclose it in quotes.

Sample entry

Here's an example entry in the **/etc/config/qtalk.modems** file:

```
hst  //1/dev/ser1 //2/dev/ser1 //2/dev/ser2
     //30/dev/ser2,""

resethst hst,"^~~~ATZ"
```

This example defines two pools: one (*hst*) a simple pool of modems; the other (*resethst*) a pool that has the same modems but with an *init_string* of **"^~~~ATZ"**. Note that one modem in the hst pool, **//30/dev/ser2**, has an empty *init_string* (i.e. **""**).

Any modems accessed through the *hst* pool won't have an *init_string* emitted to the device. But if those same modems are accessed through a reference to the *resethst* pool, all but **//30/dev/ser2** will have the **"^~~~ATZ"** *init_string* emitted to the device.

The presence of the empty *init_string* (**""**) ensures that no *init_string* will be sent to **//30/dev/ser2** regardless of whether the modem was accessed via the *hst* pool or the *resethst* pool. This happens because the *init_string* closest in the pool hierarchy to the actual naming of the device always supersedes any other *init_string*s.

As an alternative to defining the *resethst* pool, you could either name the **"^~~~ATZ"** *init_string* for each modem in the *hst* pool, or you

could specify the *init_string* with the **-m** option (**-m hst,"^~~~ATZ"** does the same as **-m resethst**).

In *init_string*s (as well as dialing strings, described below), you can specify any of the following characters; **qtalk** will act on these characters instead of sending them straight to the modem:

|	send a carriage return
~	delay for 1 second
'	delay for 100 milliseconds
^	drop **DTR** for 1 second (forces modem to hang up and reset)
!	send a 500-millisecond break
o	emit a single character represented by *o*, where *o* is an octal number of up to three digits
xhh	same as above, but specified in hex

Interactive commands

To specify any of the following commands, you first press the command character (usually Ctrl – A).

b (break) Send a break over the modem. You may also send out breaks by pressing the break key (Ctrl – Break on the console keyboard).

C (command character)

Change the command character. You will be prompted to enter the new command character in hex (e.g. 0x02) or as ^*char* (e.g. ^b).

c (change directory)

If you specify the **c** command, **qtalk** prompts you for a new directory name, and will attempt to move to that directory. If **qtalk** moves successfully to

the directory you specify, the directory becomes the current working directory (see the **pwd** utility) for the duration of **qtalk**, or until you change directory again. When **qtalk** terminates, your current working directory reverts to the directory you were in when **qtalk** was invoked.

d (dial system)

If you specify the **d** command, **qtalk** prompts you to enter a system name. It will first look for that name in your own dialing directory (**$HOME/.qtalk**) and if it doesn't find it there, it will look in the system-wide dialing directory, **/etc/config/qtalk**.

If you enter a question mark (**?**) for the system name, you'll see the contents of both dialing directories (if they exist) and you'll be prompted again for a system name. To abort the dialing command and return to normal communications mode, press Enter without entering a name.

Dialing is implemented by looking up the system name — which can also be specified on the command line when **qtalk** is invoked — first in your **$HOME/.qtalk** file, then in the **/etc/config/qtalk** file. These files share the same format:

system_name **[***dialing_string***]**
[**<***whitespace***>** *command-line_options***]**
⋮

In these files, you must specify the system name at the very left of the line with no whitespace before it. The dialing string is optional (this string contains commands to be sent to the modem before you're given interactive control). If you specify a dialing string, you must separate it from the system name by **space**s or **tab**s, or both.

In dialing strings, you can specify any of the following characters; **qtalk** will act on these characters instead of sending them straight to the modem:

|	send a carriage return
~	delay for 1 second
'	delay for 100 milliseconds
^	drop **DTR** for 1 second (forces modem to hang up and reset)
!	send a 500-millisecond break
o	emit a single character represented by *o*, where *o* is an octal number of up to three digits
\x*hh*	same as above, but specified in hex

additional lines beneath the line that defines the system name and dialing string may contain additional **qtalk** command-line options to apply when talking to this system.

If you specify these additional lines, begin each one with at least one **tab** or **space** character. Note that a single command-line option can't span multiple lines. You can, however, place just one option per line.

D (delete character)

Change the delete character. You will be prompted to enter the new delete character in hex (e.g. 0x08) or as ^*char* (e.g. **^h**).

e (echo) If you specify the **e** command, the local echo feature is toggled. Some systems expect the "terminal" to perform local echoing (half duplex).

h (hang up) If you specify the **h** command, the DTR line is lowered for approximately 1/2 a second. This

permits modems that support hardware hangup to do so.

l (log) Begin or end logging of this session. If no log file is open, **qtalk** asks for the name of a file to log into. If logging is already in progress, it is terminated and the log file is closed. The **l** command records every character that is sent or received in the log file.

You can use **l** to take "snapshots" of data from a host computer.

o (modify protocol options)

Prompts you for the protocol whose command string you wish to change (**qcp**, **ZMODEM**, **other**). The current **s**end and **r**eceive command strings are displayed and you're then prompted to enter a new command string. To set a string to null, use **""**. If you don't want to make any changes, press Enter.

p (parity) Ignore parity (top bit) of received characters. If this option is already set, turn it off.

q (quit with hangup)

This executes a hangup command (see **h** above) then causes **qtalk** to exit.

r (invoke a receive)

Invoke the currently selected file transfer protocol to receive a file. Note that **qtalk** will automatically invoke **qcp** and **ZMODEM** if it recognizes a received startup sequence of one of these protocols.

s (send a file) Send a file, using the currently selected file transfer protocol. This sends a file to another system running the same protocol, which is more reliable than simply writing the file to the modem (see the **w**rite command).

When the currently selected protocol is **qcp**, you can send more than one file by specifying the **x=***index_file* option when **qtalk** asks for the file to send. This file contains a list of files to send, one per line. You can also specify more than one filename, separated by spaces.

The **qcp** utility lets you follow the name of the file to send with the name of the destination file. You separate the two filenames with a comma. This is also true for filenames within an index file (option **x=**). For example:

```
Send file(s)?  file1  main.c,new_main.c
```

sends the file **file1** as **file1** and the file **main.c** with the name **new_main.c**. If you don't specify a new name, **qtalk** creates a file with the same name as the file that is sent.

Files received by **qtalk** using the **qcp** protocol have the same attributes and date as the file on the sending machine.

Most transfer protocols require that the modem port be configured for 8-bit data (Use the **-b** option to set the serial port to 8 data bits; also see the **stty** command).

T (switch terminal emulation)

The **T** command lets you switch the terminal emulation used by the console that **qtalk** is running on. Make sure that the device **qtalk** is running on supports the specified protocol.

You can specify one of the following terminal types:

© 2005, QNX Software Systems

This type:	Is useful for:
ansi	communicating with systems that support ANSI or VT100 type terminals. It uses ISO Latin-1 supplemental characters (default mode of **Dev.ansi** console driver).
pcansi	communicating with a DOS BBS. This type uses the IBM PC supplemental character set.
qnx	communicating between QNX systems.
qnxs	communicating with QNX2 systems that have been set up to run with a qnxs terminal.

When you use the **T** command, you'll be prompted for a terminal type. If you don't wish to set or change the terminal type, simply press Enter.

t (select transfer protocol)

Shows you which protocol is being used (e.g. **qcp**, **ZMODEM**) and asks you to select a new one. If you press Enter, no changes are made. This command controls the protocol used when you use the **s**end and **r**eceive commands from the command menu.

w (write) Write a file to the modem. The file is transmitted over the modem, and also echoed on the display if local echo is enabled. The **w**rite command doesn't perform any error checking.

You may abort writing to the modem at any time by pressing any key.

x (exit) Exit from **qtalk** without performing a hangup. It's good practice to use the **q** (quit with hangup)

command for leaving **qtalk**, unless you really don't want to perform a hangup.

! (execute a shell command)

This command lets you execute any command from within **qtalk**. You'll probably find that you'll use the **!** command often to:

- execute the **ls** command in order to see what files are in your current directory prior to sending some of them
- execute non-native file transfer protocols (such as XMODEM or Kermit) from within **qtalk** if you don't want to bother setting the other protocol to the desired commands

The ZMODEM utilities **rz** and **sz** also implement XMODEM and YMODEM.

Examples:

Call the system "home":

```
qtalk home
```

Communicate with a machine that doesn't echo (half duplex) and expects an ASCII **backspace** (08 hex) to delete characters.

```
qtalk -e -d 08
```

Communicate with another system and print a hardcopy record.

```
qtalk -l /dev/par
```

Use the **/dev/ser2** serial port on node 2.

```
qtalk -m //2/dev/ser2
```

© 2005, QNX Software Systems

Use the **qcp -n** option for **qcp** receives (i.e. receive only files that are newer than existing files).

```
qtalk -o qcp_re="qcp $MODEM re -n"
```

See also:

Dev.ansi, **qcp**, **stty**

Syntax:

Print additional memory found:

ramtrap [*options***] [query]**

Add the additional memory found to the pool of memory available to the system:

ramtrap [*options***] add**

Options:

-e *end_addr*	Scan end address. Use **0x** for hex. End in **K** (×1024) or **M** (×1024×1024). The default is end address is 128M.
-m *max*	Set scan end address to scan *start_addr* + *max*. End in **K** (×1024) or **M** (×1024×1024). The default for *max* is 128M.
-q	Suppress status messages.
-s *start_addr*	Scan start address. Use **0x** for hex. End in **K** (×1024) or M (×1024×1024). The default is to figure it out from the *osinfo* structure.
add	Add the additional memory found to the pool of memory available to the system.
query	Print the additional memory found, but do not add it to the system. (Default)

Description:

The **ramtrap** utility scans the machine it is run on for additional RAM memory which was not reported by the BIOS at the time the machine was booted. If **add** is not specified, the utility will write a list of the memory found to the standard output. If **add** is specified,

ramtrap will add the memory found to the pool of memory available to the operating system for use.

ramtrap may be run only by **root**.

Some computers may place memory for video cards directly after main memory. In this case **ramtrap** may confuse video memory with extended main memory which will crash the machine. You can use the **-s** option to step over the video memory area before scanning for additional extended memory.

Many Compaq computers require use of **ramtrap** to locate memory above 16 megabytes.

Files:

The standard input is unused.

If **-q** is not specified, a diagnostic message regarding the physical addresses being scanned will be written to the standard output. Information regarding the memory found is also written to the standard output.

If **ramtrap** is not run by **root**, or invalid command line parameters are supplied, a diagnostic message will be written to standard error and **ramtrap** will exit with a non-zero exit status.

Exit status:

>0	Could not run due to not being **root** or being supplied with invalid command-line parameters.
0	Successfully detected any available RAM (could be none)

See also:

disktrap, **inputtrap**, **mousetrap**, **nettrap**, **pcmciatrap**, **Proc**

Syntax:

rcs [*options***]** *file***...**

Options:

The **rcs** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-a*logins* Append the login names appearing in the comma-separated list *logins* to the access list of the RCS file.

-A*oldfile* Append the access list of *oldfile* to the access list of the RCS file.

-b [*rev*] Set the default branch to *rev*. If *rev* is omitted, the default branch is reset to the (dynamically) highest branch on the trunk.

-c*string* Set the comment leader to *string*. An initial **ci**, or an **rcs -i** without **-c**, guesses the comment leader from the suffix of the working filename.

This option is obsolescent, since RCS normally uses the preceding **Log** line's prefix when inserting log lines during checkout (see **co**). However, older versions of RCS use the comment leader instead of the **Log** line's prefix, so if you plan to access a file with both old and new versions of RCS, make sure its comment leader matches its **Log** line prefix.

-e [*logins*] Erase the login names appearing in the comma-separated list *logins* from the access list of the RCS file. If *logins* is omitted, erase the entire access list.

-i Create and initialize a new RCS file, but don't deposit any revision. If the RCS file has no path prefix, try to place it first into the subdirectory **./RCS**, and then into the current directory. If the RCS file already exists, print an error message.

-I Run interactively, even if the standard input isn't a terminal.

-k*subst* Set the default keyword substitution to *subst*. The effect of keyword substitution is described in **co**. Giving an explicit **-k** option to **co**, **rcsdiff**, and **rcsmerge** overrides this default. Beware of **rcs -kv**, since **-kv** is incompatible with **co -l**. Use **rcs -kkv** to restore the normal default keyword substitution.

-l [*rev*] ("el") Lock the revision with number *rev*. If a branch is given, lock the latest revision on that branch. If *rev* is omitted, lock the latest revision on the default branch. Locking prevents overlapping changes. You remove a lock with **ci** or **rcs -u** (see below).

-L Set locking to strict. Strict locking means that the owner of an RCS file isn't exempt from locking for checkin. You should use this option for files that are shared.

-m*rev***:***msg* Replace revision *rev*'s log message with *msg*.

-M Do not send mail when breaking somebody else's lock. This option is not meant for casual use; it is meant for programs that warn users by other means, and invoke **rcs -u** only as a low-level lock-breaking operation.

-n*name*[**:***rev*] Associate the symbolic name *name* with the branch or revision *rev*. Print an error message if

name is already associated with another number. If *rev* is omitted, the symbolic name is deleted.

-N*name*[**:***rev*] Act as if **-n** were specified, except override any previous assignment of name.

-o*range* Delete (i.e. "outdate") the revisions given by *range*. See "Description," below.

-q Be quiet; don't print diagnostics.

-s*state*[**:***rev*] Set the state attribute of the revision *rev* to *state*. If *rev* is a branch number, assume the latest revision on that branch. If *rev* is omitted, assume the latest revision on the default branch. Any identifier is acceptable for *state*. The following is a useful set of states:

Exp (experimental)

Stab (stable)

Rel (released)

By default, **ci** sets the state of a revision to **Exp**.

-t [*file*] Write descriptive text from the contents of the named file into the RCS file, deleting the existing text. The filename may not begin with **-**. If *file* is omitted, **rcs** obtains the text from standard input, terminated by EOF or by a line containing only a dot (**.**). You are prompted for the text if interaction is possible; see **-I**. With **-i**, descriptive text is obtained even if **-t** isn't given.

-t-*string* Write the descriptive text from *string* into the RCS file, deleting the existing text.

-T Preserve the modification time on the RCS file unless a revision is removed. This option can suppress extensive recompilation caused by a

make dependency of some copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when a change to the RCS file would mean a change to keyword strings in the working file.

-u [*rev*] Unlock the revision with number *rev*. If a branch is given, unlock the latest revision on that branch. If *rev* is omitted, remove the latest lock held by the caller.

Normally, only the locker of a revision may unlock it. Somebody else unlocking a revision breaks the lock. This causes a mail message to be sent to the original locker. The message contains a commentary solicited from the breaker. The commentary is terminated by EOF or by a line containing only a dot (**.**).

-U Set locking to non-strict. Non-strict locking means that the owner of a file need not lock a revision for checkin. You should *not* use this option for files that are shared. Whether default locking is strict is determined by your system administrator, but it is normally strict.

-V[*n*] Report RCS version; or, if a number is supplied immediately after **-V** (no spaces), emulate RCS version *n*, where *n* may be 3, 4, or 5. This option can be useful when you're interchanging RCS files with other users who are running older versions of RCS.

-x*suffixes* Use *suffixes* to characterize RCS files. See **ci** for details.

-z*zone* Use *zone* as the default time zone. This option has no effect; it is present for compatibility with other RCS commands.

Description:

The **rcs** utility creates new RCS files or changes the attributes of existing ones. An RCS file contains multiple revisions of text, an access list, a change log, descriptive text, and some control attributes.

Filenames ending in **,v** denote RCS files; all others denote working files. If a working file is given, **rcs** tries to find the corresponding RCS file first in an RCS subdirectory and then in the working file's directory, as explained for the **co** utility.

The RCS filename and the revisions outdated are written to the diagnostic output.

For **rcs** to work, the caller's login name must be on the access list, *unless* one or more of the following is true:

- The access list is empty.
- The caller is the owner of the file or the superuser.
- The **-i** option is specified.

To delete one revision or a range of revisions, you use the **-o***range* option. The following rules apply to *range*:

If *range* is:	It means:
a single revision number	that revision
a branch number	the latest revision on that branch
of the form *rev1***:***rev2*	revisions *rev1* to *rev2* on the same branch
of the form **:***rev*	from the beginning of the branch containing *rev* up to and including *rev*
of the form *rev***:**	from revision *rev* to the end of the branch containing *rev*

Note that none of the outdated revisions may have branches or locks.

Compatibility

To make an RCS file acceptable to prior RCS versions (by discarding information that would confuse those versions), use **rcs -V** *n*.

Also note the following:

- The **-b***rev* option generates an RCS file that cannot be parsed by RCS version 3 or earlier.
- The **-k***subst* options (except **-kkv**) generate an RCS file that cannot be parsed by RCS version 4 or earlier.

For more information on RCS versions, see the **co** utility.

Files:

The **rcs** utility accesses files much as **ci** does, except that it doesn't need to access the working file or its directory.

Exit status:

0 Successful completion.

>0 An error occurred.

The exit status is zero only if all operations were successful.

Author:

GNU

See also:

ci, **co**, **ident**, **rcsdiff**, **rcsmerge**, **rlog**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience 15, 7*, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly and Associates, 1995.

rcsclean

Clean up working files (UNIX)

Syntax:

rcsclean [*options***] [***file***...]**

Options:

The **rcsclean** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-k*subst* Set the default keyword substitution to *subst* when retrieving a file for comparison.

-n[*rev*] Do not actually remove any files or unlock any revisions. Using this option will tell you what **rcsclean** would do without actually doing it.

-q Be quiet; suppress diagnostic output.

-r*rev1* [**-r***rev2*]

Specify the revision for comparison.

-T Preserve the modification time on the RCS file even if the RCS file changes because a lock is removed. This option can suppress extensive recompilation caused by a **make** dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the lock removal would mean a change to keyword strings in the other working file.

-u[*rev*] Unlock the revision if it is locked and no difference is found.

-V Print the version number.

-V*n* Emulate RCS version *n*. For details, see the **co** utility.

-x*suffix*	Look at the file in **./RCS/***suffix**
-z*zone*	Set the default timezone to *zone*.
file	The pathname of a file.

Description:

The **rcsclean** command removes working files that were checked out and never modified. For each file given, **rcsclean** compares the working file and a revision in the corresponding RCS file. If it finds no difference, it removes the working file unless the working file is writable and the revision is locked. If the revision is locked, it unlocks the revision.

If no file is given, all working files in the current directory are cleaned. Any other options are passed along to **rcsdiff** for the comparison.

The **rcsclean** command is useful for clean targets in Makefiles. See also **rcsdiff**, which prints out the differences, and **ci**, which normally asks whether to check in a file if it wasn't changed.

Examples:

Remove all working files in the current directory that weren't changed since their checkout:

```
rcsclean
```

Remove all working files ending in **.c** or **.h** that weren't changed since their checkout:

```
rcsclean *.c *.h
```

Exit status:

0	No differences were found for any file under RCS control.
1	Differences were found.
2	An error occurred.

Author:

GNU

Caveats:

You can't give RCS filenames as arguments.

Any diagnostics generated by **rcsdiff** when comparing files are discarded.

If the latest revision is already unlocked, and you have a lock on an earlier revision, the earlier revision is unlocked.

See also:

ci, **co**, **ident**, **rcs**, **rcsdiff**, **rcsmerge**, **rlog**

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly and Associates, 1995.

rcsdiff

© 2005, QNX Software Systems

Compare RCS revisions (UNIX)

Syntax:

Print the version number:

```
rcsdiff -V
```

Compare RCS revisions:

rcsdiff [*options***] [***diff_options***]** *file***...**

Options:

The **rcsdiff** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-k*subst* Set the default keyword substitution to *subst*.

-q Be quiet; suppress diagnostic output.

-r*rev1* [**-r***rev2*]

Compare these revisions. Both *rev1* and *rev2* may be given numerically or symbolically.

-T Preserve the file modification time.

-V Print the version number.

-V*n* Emulate RCS version *n*. For details, see the **co** utility.

-x*suffix* Specifies the suffixes for RCS files. See **ci** for details.

-z*zone* Set the default timezone to *zone*.

diff_options Any options which apply to **diff** except **-r** may be specified to **rcsdiff**. These options will be passed on to **diff** when the revisions are being compared.

Description:

The **rcsdiff** utility runs **diff** to compare two revisions of each RCS file given. Filenames ending in **,v** (or the suffix specified by **-x**) denote RCS files; all others are assumed to be working files. The **rcsdiff** utility derives the working filename from the RCS filename and vice versa, as explained for the **co** utility. You can specify pairs consisting of an RCS filename and a working filename.

All options of **diff** that apply to regular files are accepted, with the same meaning as for **diff**.

The comparisons **rcs** performs depend on whether you specify *rev1* and/or *rev2*:

If you:	Then **rcsdiff** compares:
omit both *rev1* and *rev2*	the latest revision on the default branch (by default the trunk) with the contents of the corresponding working file. This is useful for determining what you changed since the last checkin.
specify *rev1*, but omit *rev2*	revision *rev1* of the RCS file with the contents of the corresponding working file.
specify both *rev1* and *rev2*	revisions *rev1* and *rev2* of the RCS file.

The **-k** option affects keyword substitution when extracting revisions, as described in the **co** utility. For example:

```
-kk -r1.1 -r1.2
```

ignores differences in keyword values when comparing revisions 1.1 and 1.2.

To avoid excess output from locker name substitution, **-kkvl** is assumed provided that:

- at most one revision option is given
- no **-k** option is given
- **-kkv** is the default keyword substitution
- the working file's mode would be produced by **co -l**

Examples:

Compare the latest revision on the default branch of the RCS file **RCS/f.c,v** to the contents of the **f.c.** working file:

```
rcsdiff f.c
```

Exit status:

0 No differences were found during any comparison.

1 Some differences were found.

2 An error occurred.

Author:

GNU

See also:

ci, **co**, **diff**, **diff3**, **ident**, **rcs**, **rcsmerge**, **rlog**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience 15, 7*, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly and Associates, 1995.

Syntax:

Print RCS version number:

```
rcsmerge -V
```

Merge RCS revisions:

rcsmerge [*options***]** *file*

Options:

The **rcsmerge** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-A Most verbose output. Output all changes, bracketing conflicts.

-e Output unmerged changes between revisions into working file.

-E Same as **-e**, but with brackets on conflicts.

-k*subst* Use *subst*-style keyword substitution (see **co** for details). For example, **-kk -r1.1 -r1.2** ignores differences in keyword values when merging the changes from 1.1 to 1.2.

-p[*rev*] Send the result to standard output instead of overwriting the working file.

-q[*rev*] Be quiet; don't print diagnostics.

-r[*rev*] Merge with respect to revision *rev*.

-T No effect; for compatibility.

-V[*n*]	Report RCS version; or, if a number is supplied immediately after **-V** (no spaces), emulate RCS version *n*. See **co** for details.
-x*suffix*	Get files from **./RCS/***suffix**
-z*zone*	Select time zone.
file	The pathname of a file.

Description:

The **rcsmerge** utility incorporates the changes between two revisions of an RCS file into the corresponding working file.

A filename ending in **,v** denotes an RCS file; all others are assumed to be working files. The **rcsmerge** utility derives the working filename from the RCS filename and vice versa, as explained in **co**. Note that you can specify a pair of files consisting of an RCS file and a working file.

The **rcsmerge** utility prints a warning if there are overlaps, and delimits the overlapping regions as explained in the section on "Join lists" in the **co** utility. The **rcsmerge** utility is also useful for incorporating changes into a checked-out revision.

Remember that you must specify at least one revision via options **-p**, **-q**, or **-r**. Note that can't specify more than two revisions. If you specify only one revision, the latest is omitted and the latest revision on the default branch (normally the highest branch on the trunk) is assumed for the second revision. You can specify revisions numerically or symbolically.

Examples:

Suppose you've released revision 2.8 of **f.c**. Let's also assume that after you complete an unreleased revision 3.4, you receive updates to release 2.8 from someone else. To combine the updates to 2.8 and the changes you've made between 2.8 and 3.4, you put the updates to 2.8 into file **f.c** and execute this command:

```
rcsmerge -p  -r2.8 -r3.4  f.c >f.merged.c
```

You then examine the **f.merged.c** file. Alternatively, if you want to save the updates to 2.8 in the RCS file, check them in as revision 2.8.1.1 and execute the **co -j** command:

```
ci -r2.8.1.1 f.c
co -r3.4 -j2.8:2.8.1.1  f.c
```

The following command would undo the changes between revision 2.4 and 2.8 in your currently checked-out revision in the **f.c.** file:

```
rcsmerge -r2.8 -r2.4 f.c
```

Note the order of the arguments, and that **f.c** will be overwritten.

Exit status:

0 No overlaps.

1 Some overlaps.

2 An error occurred.

Author:

GNU

See also:

ci, **co**, **diff3**, **ident**, **rcs**, **rcsdiff**, **rlog**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience 15, 7*, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly and Associates, 1995.

renice

© 2005, QNX Software Systems

Set nice values of running processes (POSIX)

Syntax:

renice *prioritylevels* **[-g** *pgrp*...**] [-p** *pid*...**] [-u** *user*...**]**

Options:

-g *pgrp*	Renice processes in this process group.
-p *pid*	Renice this process.
-u *user*	Renice processes owned by this user.
prioritylevels	The amount to adjust the priority by. Positive numbers will lower the priority by that many full priority levels, while negative numbers will raise the priority.

Description:

The **renice** utility adjusts the priority of one or more running processes. The specified *prioritylevels* is subtracted from the current priority of each selected process.

If you enter:	It will:
a positive value (e.g. **2** or **+2**)	*lower* the priority of a process, making it "nicer" to other processes
a negative value (e.g. **-2**)	*raise* the priority of a process, making it "meaner" to other processes

Users may adjust their priority as follows:

If you're:	You can change to any priority:
a normal user	from 1 to 19
a superuser (**root**)	from 1 to 29

Users without appropriate privileges can **renice** only the processes they own.

Examples:

Lower the priority of process 768 by 2:

```
renice 2 -p 768
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

nice, **slay**

rlog

© 2005, QNX Software Systems

Print log messages and other info about RCS files (UNIX)

Syntax:

rlog [*options***]** *file***...**

Options:

The **rlog** utility does not follow the standard utility syntax conventions. Options and their arguments may not be separated by white space, and options may not be combined in single command line arguments (i.e. multiple options must be separated by spaces in the command line).

-b Print information about the revisions on the default branch, normally the highest branch on the trunk.

-d*dates* Print information about revisions with a checkin date/time in the ranges given by the semicolon-separated list *dates*.

A range of the form:	**Selects:**
*d1***<**d2* or *d2***>***d1*	the revisions that were deposited between *d1* and *d2*, excluding *d1* and *d2*.
*d1***<=***d2* or *d2***>=***d1*	the revisions that were deposited between *d1* and *d2*, inclusive of *d1* and *d2*.
<*d* or *d***>**	all revisions earlier than *d*
<=*d* or *d***>=**	all revisions dated *d* or earlier
*d***<** or **>***d*	all revisions dated later than *d*
*d***<=** or **>=***d*	all revisions dated *d* or later

continued...

A range of the form:	Selects:
d	the single, latest revision dated *d* or earlier

The date/time strings *d*, *d1*, and *d2* are in the free format explained in **co**. Quoting is normally necessary, especially for **<** and **>**.

-h Print only the RCS filename, working filename, head, default branch, locks, access list, symbolic names, keyword substitutions and total revisions.

-l[*lockers*] ("el") Print information about locked revisions only. If *lockers*, a comma-separated list of login names, is given, ignore all locks other than those held by the named lockers. For example, this command:

```
rlog -L -R -lwft RCS/*,v
```

prints the name of RCS files locked by the user **wft**.

-L Ignore RCS files that have no locks set. This is useful in combination with options **-R**, **-h**, and **-l**.

-N Do not print symbolic names.

-t Act as if **-h** were specified, but also print the descriptive text.

-r*revisions* Print information about revisions given in the comma-separated list *revisions* of revisions and ranges. For more information, see the Description, below.

-R Print only the name of the RCS file. This is useful for translating a working filename into an RCS filename.

-s*states* Print information about revisions whose state attributes match one of the states given in the comma-separated list *states*.

© 2005, QNX Software Systems

-T	This option has no effect; it is present for compatibility with other RCS commands.
-w[*logins*]	Print information about revisions checked in by users with login names appearing in the comma-separated list *logins*. If *logins* is omitted, the user's login is assumed.
-V[*n*]	Report RCS version; or, if a number is supplied immediately after **-V** (no spaces), emulate RCS version *n* when generating logs, where *n* may be 3, 4, or 5. This option can be useful when you're interchanging RCS files with other users who are running older versions of RCS (see also "Determining RCS version" in **co**).
-x*suffixes*	Use *suffixes* to characterize RCS files. See **ci** for details.
-z*zone*	specifies the date output format, and specifies the default time zone for dates in the **-d***dates* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication.

Description:

The **rlog** utility prints information about RCS files. Filenames ending in **,v** denote RCS files; all others are assumed to be working files. If a working file is given, **rlog** tries to find the corresponding RCS file first in an **RCS** subdirectory and then in the working file's directory, as explained in **co**.

The **rlog** utility prints the following information for each RCS file:

- RCS filename

- working filename
- head (i.e., the number of the latest revision on the trunk)
- default branch
- locks
- access list
- symbolic names
- suffix
- total number of revisions
- number of revisions selected for printing
- descriptive text

This information is followed by entries for the selected revisions in reverse chronological order for each branch. For each revision, **rlog** prints:

- revision number
- date/time
- author
- state
- number of lines added/deleted with respect to the previous revision
- locker of the revision (if any)
- log message

All times are displayed in UTC. Without options, **rlog** prints complete information. Options restrict this output.

The **rlog** utility prints the intersection of the revisions selected with the options **-d**, **-l**, **-s**, and **-w**, intersected with the union of the revisions selected by **-b** and **-r**.

© 2005, QNX Software Systems

When using the **-r***revisions* option, note that the following rules apply to the *revisions* argument:

If you specify:	Then **rlog** prints info about:
a range *rev1***:***rev2*	revisions *rev1* to *rev2* on the same branch
:*rev*	revisions from the beginning of the branch up to and including *rev*
*rev***:**	revisions starting with *rev* to the end of the branch containing *rev*
a branch	all revisions on that branch
a range of branches	all revisions on the branches in that range

Examples:

The first of the following commands prints the names of all RCS files in the subdirectory **RCS** that have locks. The second command prints the headers of those files, while the third prints the headers plus the log messages of the locked revisions. The last command prints complete information.

```
rlog -L -R RCS/*,v
rlog -L -h RCS/*,v
rlog -L -l RCS/*,v
rlog RCS/*,v
```

Exit status:

0 Successful completion.

>0 An error occurred.

The exit status is zero only if *all* operations were successful.

Author:

GNU

See also:

ci, **co**, **ident**, **rcs**, **rcsdiff**, **rcsmerge**

Walter F. Tichy, "RCS — A System for Version Control," *Software — Practice & Experience 15, 7*, July 1985.

Don Bolinger & Tan Bronson, *Applying RCS & SCCS*, O'Reilly and Associates, 1995.

Remove files (POSIX)

Syntax:

rm [-Rfir] [-d] [-l *n***] [-v]** *file***...**

Options:

-d (QNX extension) If the **-R** option is specified, remove the files but leave the directory tree intact (i.e. no **rmdir** is performed).

-f Force each specified file to be removed without prompting for confirmation.

-i Be interactive; request confirmation before removing each existing file.

-l *n* ("el") (QNX extension) If the **-R** option is specified, recurse only *n* levels down a directory tree.

-r Equivalent to the **-R** option (below).

-R Recursively remove files and subdirectories under directories given as arguments. This process removes the directory and the entire file tree under it.

CAUTION: The **-R** option must be used with care, as it removes directories, subdirectories, and files. Using the **-i** option with **-R** adds a measure of safety by turning on interactive prompting before each file or directory is removed.

-v (QNX extension) Be verbose; print out files and directories as they are removed.

file The pathname of a file to be removed.

Description:

The **rm** utility removes each specified file from a directory.

By default, **rm** refuses to remove any file that names a directory. This may be overridden with the **-R** or **-r** options. In any case, **rm** always refuses to remove the current working directory.

If a *file* operand has been specified but doesn't exist and the **-f** option hasn't been given, a message is written to the standard error output. If **-f** has been given, the error message isn't written. In either case, **rm** goes on to any remaining *file*s.

The **rm** utility doesn't necessarily remove the file data from the disk. A file may have more than one link; that is, it may be known by more than one name in the filesystem (see the **ln** utility for information on creating links). The **rm** utility breaks one such link; it dissociates the file from one name. If this is the only link, the file data become inaccessible and the file space is returned to the system for reuse. Otherwise, the data remains accessible via other names.

Examples:

Remove the **a.out** and **core** files.

```
rm a.out core
```

Remove the directory **junk** and all its contents, without prompting.

```
rm -Rf junk
```

Environment variables:

If **POSIX_STRICT** is set, **-f** and **-i** will be operationally exclusive — only the last instance of **-i** or **-f** will take effect; instances appearing earlier on the command line will be ignored. i.e. **rm -i -f *** = force, **rm -f -i *** = interactive. If **POSIX_STRICT** is not set, **-i** will require confirmations regardless of whether or not **-f** is specified.

In a future release, **rm**'s default behavior will be changed to the POSIX behavior. To work correctly with both current and later versions, either avoid the use of **-f** and **-i** together, or run **rm** with **POSIX_STRICT** set.

Exit status:

0 All the named files were removed.

>0 An error occurred.

See also:

find (**-remove!** primitive), **rmdir**

rmdir

Remove directories (POSIX)

Syntax:

rmdir [-p] *dir*...

Options:

-p Remove the entire directory paths (consisting only of directories which are empty except for the named pathname components) of the directories specified on the command line. (See example)

dir The pathname of an empty directory to be removed.

Description:

The **rmdir** utility removes the directory specified by each *dir* operand, provided the directory is empty.

The **rmdir** utility processes directories in the order they're specified on the command line. If you specify a parent directory, you should specify its subdirectory *before* the parent directory. That way, the parent directory will be empty when the **rmdir** utility tries to remove it.

Examples:

Remove the subdirectory **oldfiles** from the **/home/fred** directory:

```
rmdir /home/fred/oldfiles
```

Remove the directory path **dir1/dir2/dir3**:

```
rmdir -p dir1/dir2/dir3
```

For this command, **rmdir** first removes **dir1/dir2/dir3**. If this is successful, it then removes **dir1/dir2**. If this is also successful, it removes **dir1**.

© 2005, QNX Software Systems

Exit status:

0 Each directory specified by a *dir* operand referred to an empty directory and was removed successfully.

>0 An error occurred.

See also:

find (**-remove!** primitive), **rm**

Syntax:

romqnx [options] *buildqnx_image rom_image*

Targets:

QNX 4

Options:

-c	Compress the RAM portion.
-d *module#*	Force this entire module into RAM. A value of 0 means all modules.
-R *addr*	The EPROM/Flash/SRAM address (the default is 15M).
-r *addr*	The RAM address (the default is 4K).
-v	Be verbose.

Description:

The **romqnx** utility takes an OS image produced by **buildqnx** and modifies it to make it suitable for placing in ROM or Flash. The resulting image consists of code and data for each module in the original buildfile. You must always copy the data portion of each module to RAM. You may execute the code portion in place (XIP) out of ROM or Flash, or you may copy it to RAM with the data.

If your ROM or Flash is bank-switched, you must copy all code to RAM. You can execute only out of linearly mapped ROM or Flash.

Since you'll always be copying the data, it's advantageous to compress it in ROM or Flash and expand it into RAM. To do this, use the **-c** option. This approach typically results in significant Flash/EPROM savings.

Systems that can execute out of ROM or Flash often pay a performance penalty. It's not uncommon for ROM or Flash to have extra wait states or a narrow bus interface (8 or 16 bit). By default, **romqnx** sets up the image to execute out of ROM or Flash; by using the **-d** option, you can force any module to be copied to RAM. If your ROM or Flash is bank-switched, you must specify **-d 0** to force all modules into RAM. If you build the system debugger (**/boot/sys/Debugger**) into the image, the debugger is always copied to RAM.

The **romqnx** utility needs to know the address of the ROM where the image will be placed and the address of the RAM where data (or code) will be copied. The default is 15M (0xF00000) for ROM and 4K (0x1000) for RAM.

These addresses must be multiples of 4096 (0x1000), which is the page size of the processor.

If you specify **-d 0** to copy all modules into RAM, the **-R** *addr* option is ignored.

Examples:

Build a ROM image for an embedded system in which the code runs out of ROM and the data is decompressed from ROM into RAM. The ROM is at 63M and the OS RAM data starts at the default address of 4K:

```
cd /boot
buildqnx build/386ex images/386ex
romqnx -c -R 0x3f00000 images/386ex images/386ex.rom
```

The same as above except that the Process Manager and Kernel are copied into RAM, which may be faster that the ROM:

```
cd /boot
buildqnx build/386ex images/386ex
romqnx -c -R 0x3f00000 -d 1 images/386ex images/386ex.rom
```

See also:

buildqnx

rtc

© 2005, QNX Software Systems

Set or get date from realtime clock (QNX)

Syntax:

Update the current time based on the time from the specified clock:

rtc [-l] [-r *rate***] [-S** *seconds***]** *clock_type*

Set the time of the specified clock to the current time:

rtc -s [-l] *clock_type*

Options:

-l ("el") Set hardware time to local time, not Coordinated Universal Time (*UTC*). UTC is the standard term for Greenwich Mean Time (*GMT*). **-l** has no effect if the **clock_type** is **net**.

-r *rate* Rate at which to adjust the OS time if it is currently within 60 seconds (or value set by **-S**) of the time from the source specified by *clock_type*. The *rate* is turned into a ratio 1/rate. The default rate is 100 (1/100 = 1%).

-s Set hardware to current date and time.

-S *seconds* Specify maximum number of seconds difference between current time and new time before jam loading rather than slowly adjusting. (see **-r** for the convergence speed). Default: 60.

clock_type One of the following:

Clock type	Description
hw	hardware clock (automatically selects one of **at** or **ps2**)

continued...

Clock type	Description
net [*node*]	a node on a QNX network (node 1 is used unless otherwise specified)
at	(deprecated) IBM PC/AT Compatible hardware clock
ps2	(deprecated) IBM PS/2 Compatible hardware clock

Description:

The **rtc** command gets or sets the date and time from a battery backed-up hardware clock or from another QNX node.

If your machine has a built-in clock/calendar, you should include the following command in your **/etc/config/sysinit** file:

```
rtc hw
```

If the time in your hardware clock is incorrect (perhaps the battery has been replaced), you can set the time using the **-s** (set) option. First, set the QNX time with the **date** utility. Then, issue the appropriate **rtc** command. For example:

```
rtc -s hw
```

If the **Proc** manager on your machine has recognized a new machine type that your version of **rtc** does not know about, **rtc hw** will fail. If this happens, contact QNX Software Systems to let them know about the problem, and work-around it until a new version of **rtc** is made available by using one of the deprecated overrides i.e. specifying **at** or **ps2** instead of **hw**.

You can use clock type **net** [*node*] to get the date from a specified *node*, or to set the date on a specified *node*. If *node* isn't specified, the default is node 1. When clock type **net** [*node*] is used, the **-l** option has no effect.

Examples:

Update the current date and time from the hardware clock:

```
rtc hw
```

Set hardware clock with current date and time:

```
rtc -s hw
```

Update the current date and time on the local node from the current date and time on node 1:

```
rtc net
```

Update the current date and time on the local node from the current date and time on node 5:

```
rtc net 5
```

Set current date and time on node 5 to the current date and time on the local node:

```
rtc -s net 5
```

Exit status:

0 Successful.

>0 An error occurred.

Caveats:

Past releases of the **rtc** utility had varying behavior with respect to the year 2000 and the CMOS century byte. As of the printing of this document, the **rtc** program neither utilizes nor sets the century byte in the CMOS RAM. However, the BIOS and other operating systems may use the century byte and many BIOSes fail to handle the year

2000 rollover correctly. If you use the same machine for both QNX and other operating systems, contact your BIOS manufacturer and/or operating system vendor for information on any procedures that need to be followed to assure correct operation of those products.

Detailed information on the year 2000 as it pertains to current and past releases of the QNX operating system is available in a product bulletin on the QNX web site (**http://www.qnx.com**).

See also:

clock, **date**

sac

© 2005, QNX Software Systems

Display system activity at each priority level (QNX)

Syntax:

sac [-i *factor***] [-n** *node***] [-p** *priority***] [-s** *seconds***]**
[<*file***] [>***file***]**

Options:

-i *factor*	Use this integration factor (0 to 12; disable = 0).
-n *node*	Gather statistics on *node* (default is the local node).
-p *priority*	Run at this priority (default is 10).
-s *seconds*	Use this interval between snapshots (default is 1.0).
<*file*	Run previously saved log file.
>*file*	Save data to a log file.

Description:

The **sac** utility is a system activity monitor. It displays a bar graph showing the amount of processor activity at each priority level. If you run **sac** on an idle system, you would see one large bar at priority 0 (priority 1 if **clock** is running).

The Kernel collects statistics on processor activity at each priority level. These statistics are integrated over time. The **-i** option sets the integration factor for the system. Smaller values average over shorter periods, resulting in a faster response to changes in system activity. Larger values let you monitor average activity over a longer period of time. If you set the integration factor to 0, then statistics won't be gathered. If you don't specify the **-i** option, then the current integration factor will be used. If a node is not gathering statistics, you will be told to set a value for the integration factor. In this case, re-run **sac** with a nonzero value for the **-i** option.

Note that the **-i** option doesn't affect process scheduling. It affects only the statistics gathered for display.

By default, **sac** takes a snapshot of the system statistics once every second. You can adjust this using the **-s** option, which takes an argument in seconds and fractions of seconds. For example, **-s 0.5** would snap every half second. To stop **sac** after the next snap, press Enter.

If you specify a file for the standard output, **sac** will write a log of system activity to the file. You may play the log back by invoking **sac** with its standard input redirected.

Examples:

Take a snapshot of an active system with an integration factor of 8:

```
sac -i 8
```

Start a realtime monitor of the dynamics of the system:

```
sac -i 3 -s .1
```

Monitor node 8 using its current integration factor:

```
sac -n 8
```

Caveats:

On slow machines, **sac** may itself affect priority loading when it runs.

Note that playback data saved and re-displayed using standard output and standard input are *not* timestamped. If identical **-s** option settings are not used for both the recording and playback invocations of **sac**, the timing of the playback may vary greatly from the timing of the original data. Additional variation in playback versus recording can occur due to variations in the time taken to write the data versus the

time taken to read the data back. For this reason, playback should be used only to show what sequence of events occurred and should not be relied on to indicate the exact timing of those events.

See also:

sin, **ps**

Syntax:

scsi_util [-n *node***] [[***device***]** *command***]...**

Options:

-n *node* Specify node where the device to be controlled resides (not necessary if a full path is supplied as the optional *device* argument).

device The pathname (either full, or basename portion only) of the device to be manipulated.

command A command to perform on the selected device. For SCSI, and ATAPI or EIDE devices controlled by **Fsys.eide**, the command may be:

scan	Scan for all devices in **/dev** (default)
error	Detail on last error logged by the driver for this device
lock	Lock the media in the device
unlock	Unlock the media in the device
start	Start unit (spin up device)
stop	Stop unit (spin down)

and in addition, the following commands are available for SCSI devices only:

'defect [*option*]**'**

Display defect list for the device. *option* may be one of:

© 2005, QNX Software Systems

Option	Description
g or **G**	Display grown defect list
p or **P**	Display primary defect list

Both lists are displayed if no option is specified. Note that the **defect** command and its option must be specified as a single command-line parameter to **scsi_util**. This can be done by the use of quotes on the command line (shown), or by putting a **,** or **=** character between the command **defect** and the option e.g. **defect=g** or **defect,g**.

'density [*code*]**'**

Display or change density for a device. The density code is unique for each device type. Refer to the mode parameters for this device type in the SCSI spec. If no option is specified the current density is displayed. Note that the **density** command, like the **defect** command, must be specified with its option as a single command-line parameter to **scsi_util**, by the use of quotes on the command-line (shown) or separating the command from the code with a **,** or **=** character instead of a space.

eject Eject the media from the device

info Report basic SCSI information i.e. host adapter ID, target ID, LUN ID, and device type.

inquiry Issue the SCSI *inquiry* command to the device. For a detailed description of the data returned, refer to the SCSI specification.

'medium [*type*]**'**

Display or change medium type for a device. The medium types are unique for each device type. Refer to the mode parameters for the particular device type in the SCSI spec. If no type is specified the current type is displayed. The **medium** command and the type must be specified as a single command-line option to **scsi_util**, through use of command-line quoting (shown) or by placing a **,** or **=** between the command and the type.

sense Issue a SCSI *mode sense* command asking for current values of all mode pages. For a detailed description of the data returned, see the SCSI specification.

The first *command* must be preceded by a *device*. For subsequent *commands* the previous device will be used, or a new device may optionally be specified.

Description:

The **scsi_util** utility is used to control SCSI devices that require manipulation beyond what is afforded by the standard file utilities. Such devices and read/write optical drives and removable hard disks often must be locked, unlocked, started and stopped by this utility.

Typically the permissions on the device will be such that you will need to be **root** to run **scsi_util**.

Examples:

Scan **/dev** for SCSI devices:

```
scsi_util
```

Stop, unlock, and eject a removable SCSI device **/dev/hd1**:

© 2005, QNX Software Systems

```
scsi_util /dev/hd1 stop unlock eject
```

Files:

All output is written to the standard output, except when an *ioctl()* fails, in which case a diagnostic message is written to standard error.

Exit status:

The **scsi_util** utility always exits with a status of 0.

See also:

Blkfsys, **Fsys**, **Fsys.***

sed

Stream editor (POSIX)

Syntax:

sed [[-n] *script* **[***file***...]**

sed [-n] [-e *script***]... [-f** *script_file***]... [***file***...]**

Options:

-e *script*	Uses the command-line *script* for editing *file*s. If you specify multiple **-e** options, the *script*s are applied in the order specified to each line of the input files. If a **-f** option is specified in addition to **-e**, lines are acted upon by *script*s first.
-f *script_file*	Use the file *script_file* as the script of editing instructions. If multiple **-f** options are specified, the *scripts* are applied in the order specified to each line of the input files. If a **-e** option is specified in addition to **-f**, lines are acted upon by *script*s first.
-n	Suppress the default output, which passes each line to standard output after it's examined for editing. Only lines explicitly selected for output are written.
file	The pathname of a text file whose contents are read and edited. If you specify multiple files, the files are read in the order specified and the concatenation is edited. If no files are specified, the standard input is used.
script	A *script* consists of one or more editing instructions that you enter on the command line.

Description:

The **sed** utility is a stream editor that reads one or more text files, makes editing changes according to editing commands that you specify, and writes the results to standard output.

© 2005, QNX Software Systems

The **sed** commands are usually stored in a program file (*script_file*), although you may give simple **sed** commands from the command line. By default, **sed** copies files from the file list to its standard output, editing the lines in the process. Conceptually, there is *one* input file, which is the concatenation of all input files specified.

Lines are selected for editing based on their position within the input file, or by pattern matching. If no files are listed, input is taken from standard input (this is the only time standard input is used). The **sed** utility initially reads all the editing commands from all specified sources and places them in an internal table in the order specified. The utility then processes the (concatenated) input file as follows:

1. Read one line from the input file and copy to the *pattern space* (a work area).
2. For each command that selects the line, act on the pattern space as the edit command specifies, cyclically placing the result into the pattern space.
3. After all commands have been checked against the pattern space, write the pattern space to the standard output, provided **-n** was *not* specified. The pattern space may contain zero, one, or several lines at this time.
4. Delete the contents of the pattern space.
5. Repeat from step 1 until all of the (concatenated) input file has been read and processed.

Scripts

A *script* consists of editing commands, one per line, of the following form:

[*address*[**,** *address*]]*function*[*arguments*]

The *script* of editing commands can be given in the command line, or can be contained in the file *script_file*.

In default operation, **sed** cyclically copies a line of input into a pattern space (unless there is something left after a **D** command), applies in sequence all commands whose addresses select that pattern space, and at the end of the script copies the pattern space to the standard output (except under **-n**) and deletes the pattern space.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval. The *pattern* and *hold spaces* are each limited to 20K.

Addresses

An address is one of the following:

- a decimal number that counts input lines cumulatively across files
- a **$** token that addresses the last line of input
- a context address
- a regular expression

A command line with no address selects every pattern space. A command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second address. (If the line number of the second address is less than or equal to the line number first selected, then only the first line is selected.) Starting at the first line following the selected range, **sed** looks again for the first address. Thereafter the process is repeated.

Editing commands can be applied to non-selected pattern spaces by use of the negation character (**!**). For more information, see the section on “Editing commands.”

© 2005, QNX Software Systems

Regular expressions

The **sed** utility uses basic regular expressions (*RE*s), with the following additions:

- In a context address, the construction *****?*RE*?* (where *?* is any character) is mapped to **/***RE***/**. Note that in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the regular expression is **abcxdef**.
- The escape sequence **\n** matches a **newline** embedded in the pattern space.
- A period (**.**) matches any character except the last **newline** of the pattern space.

Editing commands

In the following list of functions, the maximum number of permissible addresses for each function is indicated by one of [*0addr*], [*1addr*] or [*2addr*] representing a maximum of zero addresses, one address or two addresses respectively. The argument *text* consists of one or more lines. Each embedded **newline** in the text must be preceded by a backslash. Other backslashes in text are treated like the backslashes in the replacement string of an **s** editing command, and can be used to protect initial **blank**s against the stripping that is done on every script line.

The **r** and **w** editing commands take an optional *rfile* (or *wfile*) parameter, separated from the command letter by zero or more **blank**s.

The arguments *rfile* or *wfile* terminate the command line. Each *wfile* is created before processing begins. There can be at most ten distinct *wfile* arguments in the script.

The **b**, **r**, **s**,**t**, **w**, **y**, **!**, and **:** editing commands take additional arguments. The following synopses indicate which arguments are separated from the commands by **blank**s:

[*2addr*] **{** *command_list*
}

Executes *command_list* only when the pattern space is selected. (Note that the trailing **}** must be the first non-blank character in the line.)

[*1addr*]**a**
text

Writes *text* to the standard output after the pattern space is written.

[*2addr*]**b** *label*

Branches to the : (colon) command bearing the *label*. If *label* is empty, branch to the end of the script.

[*2addr*]**c**
text

Deletes the pattern space. With 0 or 1 address or at the end of a 2-address range, places *text* on the output.

[*2addr*]**d**

Deletes the pattern space and starts the next cycle.

[*2addr*]**D**

Deletes the initial segment of the pattern space through the first **newline** and starts the next cycle.

[*2addr*]**g**

Replaces the contents of the pattern space by the contents of the hold space.

[*2addr*]**G**

Appends the contents of the hold space to the pattern space.

[*2addr*]**h**

Replaces the contents of the hold space by the contents of the pattern space.

[*2addr*]**H**

Appends the contents of the pattern space to the hold space.

[*1addr*]**i**
text

Writes *text* to the standard output before the pattern space is written.

© 2005, QNX Software Systems

[*2addr*]**l** Lists the pattern space on the standard output in an unambiguous form. Nonprinting characters are listed as hexadecimal digit pairs, with a preceding backslash, with the following exceptions:

Character	Listed as
alert	**\a**
backslash	****
backspace	**\b**
carriage return	**\r**
form-feed	**\f**
newline	**\n**
tab	**\t**
vertical tab	**\v**

Long lines are folded; the length at which folding occurs is unspecified, but should be appropriate for the output device.

[*2addr*]**n** Copies the pattern space to the standard output and replaces the pattern space with the next line of input.

[*2addr*]**N** Appends the next line of input to the pattern space, using an embedded **newline** to separate the appended material from the original material. Note that the current line number changes.

[*2addr*]**p** Copies (prints) the pattern space to the standard output.

[*2addr*]**P** Copies (prints) the pattern space, up to the first **newline**, to the standard output.

[*1addr*]**q** Branches to the end of the script and quits without starting a new cycle.

[*1addr*]**r** *rfile* Reads the contents of the *rfile* file. The contents are placed on the output before reading the next input line.

[*2addr*]**s/***regular expression***/***replacement string***/***flags*

Substitutes the *replacement string* for instances of *regular expression* in the pattern space. Any character can be used instead of **/**. The value of *flags* is zero or more of:

n	*n*=1 to 512. Substitutes for the *n*th occurrence only of the *regular expression* found within the pattern space.
g	Globally substitutes for all non-overlapping instances of the *regular expression* rather than just the first one. If both **g** and **n** are specified, **g** takes precedence.
p	Prints the pattern space if a replacement was made.
w *wfile*	Appends (writes) the pattern space to *wfile* if a replacement was made.

[*2addr*]**t** *label* Tests. Branches to the **:** (colon) command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a **t**. If *label* is empty, branches to the end of the script.

[*2addr*]**w** *wfile* Appends (writes) the pattern space to *wfile*.

[*2addr*]**x** Exchanges the contents of the pattern and hold spaces.

[*2addr*]**y/***string1***/***string2***/**

Replaces all occurrences of collating elements in *string1* with the corresponding collating element in *string2*. The lengths of *string1* and *string2* should be equal.

[*2addr*]**!** *function*

Applies the *function* (or command list, if *function* is **{**) only to the lines that aren't selected by the addresses.

[*0addr*]**:***label* This command does nothing; it bears a *label* for the **b** and **t** commands to branch to.

[*1addr*]**=** Places the current line number on the standard output as a line with its own line number.

[*0addr*] An empty command is ignored.

[*0addr*]**#** If a **#** appears as the first character on any line of a script file, that entire line is ignored (treated as a comment), with the single exception that if the first line of the script file begins with **#n**, the default output is suppressed (as when the **-n** option is specified on the command line).

Examples:

In the file **myfile**, find and output only those lines containing the string "**tom**":

```
sed -n -e "/tom/p" myfile
```

In the file **myfile**, replace all occurrences of the string **beneath** with the string **below**, and output to the file **newfile**:

```
sed -e "s/beneath/below/" myfile >newfile
```

Files:

All files are text files. The *script_files* named by the **-f** option consist of editing commands, one per line. Any number of additional text input files may be specified by the **r** command for insertion of *unedited* data to the standard output at points predetermined by editing rules. A maximum of 10 additional output files may be specified through the use of the **w** command in the script.

Exit status:

0 Successful completion.

>0 An error occurred.

Errors:

If one or more of the input files (this doesn't include script files) can't be opened for reading, **sed** continues to process the remaining files.

Author:

NetBSD; Diomidis Spinellis.

Copyright notice

Copyright (c) 1992 Diomidis Spinellis.

Copyright (c) 1992, 1993
The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Diomidis Spinellis of Imperial College, University of London.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1 Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following

disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

See also:

awk

Dale Dougherty, *sed & awk*, O'Reilly and Associates, 1990.

Brian W. Kernighan and Rob Pike, *The UNIX Programming Environment*, Prentice-Hall, 1984.

Syntax:

sh [-aefhkmnquvx] [*command_file* **[***argument***...]]**

sh -c [-aefhkmnquvx] *command_string*
[*command_name* **[***argument***...]]**

sh -i [-aefhkmquvx] [*command_file* **[***argument***...]]**

sh -s [-aefhkmnquvx] [*argument***...]**

Options:

By default, the Shell is invoked interactively. To change how the Shell executes shell commands and where it reads them from, you use these options:

-c Read commands from *command_string*. The special parameter $*0* is set to *command_name* and the positional parameters ($*1*, $*2*... $*n*) are set to the remaining *arguments*.

-i Run interactively. The Shell is also invoked interactively if no operands are specified and the Shell's standard input and standard error are attached to a console or terminal.

-s Read commands from the standard input. If no operands are given and the **-c** option isn't specified, the Shell is invoked as if the **-s** option had been specified.

For information on the options **-a**, **-e**, **-f**, **-h**, **-k**, **-m**, **-n**, **-q**, **-u**, **-v**, and **-x**, see the description of the **set** utility in the section "Using Shell commands."

Description:

The **sh** utility is a command interpreter that acts as the interface between the user and QNX. A clone of the 86 KornShell (**ksh86**), which is a superset of the Bourne Shell, the **sh** utility has many capabilities. These capabilities are described throughout the following sections, arranged alphabetically:

© 2005, QNX Software Systems

- Command lists
- Command substitution
- Environment variables
- Expansion
- Expressions
- Input line editing
- Initialization files
- Pipes
- Quoting
- Redirection
- Remote execution
- Reserved words
- Shell scripts
- Signals
- Using Shell commands

At the end of this man page you'll find several example shell scripts.

Terminology

The following terms have special meanings in the context of **sh**. Becoming familiar with them will help you take full advantage of the sections that follow.

AND-OR-list — A sequence of one or more pipelines separated by the operators **&&** and **||**.

blank
: Any one of the characters currently set in the **IFS** (Input Field Separators) environment variable. The defaults are usually the **tab**, **space** and **newline** characters.

command list
: A sequence of one or more **AND**-**OR**-lists separated by the operators **;** and **&**.

comment
: Any word beginning with a **#** character. The word and all characters following it are ignored until a **newline** is detected.

control operator
: One of the following symbols:

```
&  &&  (  )  ;  ;;  |  ||
```

or a **newline** character.

name
: A sequence of letters, digits, or underscores beginning with a letter or an underscore.

parameter
: A name, a digit, or any of these characters:

```
@  #  ?  -  $  !
```

If a named parameter is exported, it becomes an environment variable.

pipeline
: A sequence of one or more commands separated by the control operator **|**. The standard output of all but the last command is connected to the standard input of the next command.

redirection operator
: One of the following symbols:

```
<  >  >|  <<  <<-  >>
<&  >&  <>  <&-  >&-
```

simple command
: A sequence of optional variable assignments and redirections, optionally followed by words and redirections, and terminated with a control operator.

word
: Any sequence of characters — other than a control operator or redirection operator — that the Shell treats as a single unit when reading input.

Command lists

A *command list* is a sequence of commands or pipelines, or both. You can separate the components in a command list by either of these control operators:

`;` (used for sequential lists)

`&` (used for background execution)

or by either of these logical control operators:

`&&` (used for **`AND`** lists)

`||` (used for **`OR`** lists)

Sequential lists

If you separate the commands in a list with the `;` control operator, the commands are executed sequentially, one after the other. For example:

```
$ cd /tmp ; ls ; ls | wc -l
doit.awk     junk        temp        xfile
4
$
```

This is identical to entering the commands on three separate lines except that the Shell outputs only a single prompt once all the commands have finished executing.

Background lists

If you terminate a single command with the **&**, the command is executed in the background (asynchronously). You can also use this technique to execute a grouped command list in the background:

```
(cd /tmp; ls; ls | wc -l) &
```

For more information, see "Grouping command lists," below.

AND/OR lists

The **&&** and **||** operators allow a list to execute depending on the exit code of the preceding list:

If you use:	The next list is executed only if:
&&	the preceding list succeeded (zero exit code)
||	the preceding list failed (nonzero exit code)

Grouping command lists

You can group command lists as follows:

(*list***)** Executes *list* in a subshell.

{*list;***}** Executes *list* in the current process environment.

For example, the following group of commands will invoke a subshell that outputs a sorted list of who is on the network:

```
( who -a | cut -f1 -d' ' | sed -e 1d | sort -u )
```

© 2005, QNX Software Systems

Command substitution

Command substitution allows the standard output of a command to be substituted in place of the command itself. You can perform command substitution in either of two ways:

- by enclosing the command in backquote characters:

  ‘*command*‘

- by enclosing the command in parentheses and preceding it with a dollar sign:

  $(*command***)**

Any trailing sequence of one or more **newline** characters is deleted before the output is substituted.

Examples

The command **echo $(date)** could produce:

```
Thu Apr 11 10:23:12 EDT 1997
```

If placed in a shell script, the following command lets you find the name of the script:

```
name=‘basename -- $0‘
```

Environment variables

Environment variables are named parameters used by the Shell. You can change these variables to modify the shell environment. When invoked, the Shell sets default values for the **IFS**, **PATH**, **PS1**, and **PS2** variables.

CDPATH	The directory search path used by the **cd** builtin.
HOME	The pathname of the user’s home directory

IFS	(**I**nput **F**ield **S**eparators) A string treated as a list of characters that are used for splitting fields. The default characters in **IFS** are **`space`**, **`tab`**, and **`newline`**.
LOGNAME	The user's **`login`** name.
PATH	The directory search path used by the Shell for locating executable programs and shell scripts. For non-superusers, the default path is **PATH** is **`:/bin:/usr/bin`**
PS1	The primary prompt string (default is "$ " for non-**`root`** users, "# " for **`root`**).
PS2	The secondary prompt string (default is "**`>`** ").
SHELL	The pathname of the user's preferred shell.
TERM	The terminal type.
TMPDIR	The pathname of a directory that can be used by utilities that need a place to create temporary files.
TZ	The timezone setting. See also the section on "Named parameters" in the "Expansion" entry.

Expansion

Filename expansion

You use most QNX commands for manipulating files in one way or another. As such, the Shell has a filename "shorthand" that can be used to specify the files that a particular command is to operate on. This "shorthand" consists of these characters:

`*`

`?`

`[]`

If one or more of these shorthand characters is used in a command line, each word containing the characters is treated as a "pattern" and is replaced with the list of filenames matching the pattern. Filenames that begin with a dot (**.**) aren't included in the generated list unless a dot is explicitly included as the first character of the pattern.

*****	Matches any string, including the null string.
?	Matches any single character.
[...]	Matches any one of the enclosed characters. If you specify two characters separated by a dash (**-**), this matches the characters and any characters that fall between them (e.g. **[a-c]** matches "**a**", "**b**", and "**c**").
[!...]	Matches any character except for the enclosed characters.

Note that directories, as well as files, are matched by these shorthand characters.

Examples

The following example copies all files consisting of "**fred**" followed by a single digit from **1** to **3** (i.e. **fred1**, **fred2**, **fred3**) to the **/tmp** directory:

```
cp fred[1-3] /tmp
```

This next example copies all files ending with **.c** or **.h** (e.g. **fred.c**, **barn.h**, etc.):

```
cp *.[ch] /tmp
```

This next example copies all files that *don't* end with **.o**:

```
cp *.[!o] /tmp
```

Parameter expansion

The QNX Shell supports three kinds of parameters:

- positional parameters
- special parameters
- named parameters (or environment variables)

Parameter expansion is performed on words beginning with **$** and enclosed in braces (**{}**). The word needn't be enclosed in braces if it's used in an unambiguous context.

Positional parameters

Positional parameters are the parameters specified on the command line of the Shell and are accessed using the corresponding ordinal number. That is, $*1* expands to the first argument, $*2* expands to the second, and so on. Positional parameters can be assigned values using the special builtin command **set**.

Special parameters

The following special parameters are automatically set by the Shell and provide some commonly desired functionality for shell programming. The special parameters and the values to which they expand are as follows:

$* All the positional parameters starting from one. If the expansion occurs within double quotes, a single word with the value of each parameter separated by a **space** is generated. If the named parameter **IFS** is set, the words are separated by the first character of its value. (For more information, see the entry on "Environment variables.")

$@ All the positional parameters starting from one. If the expansion occurs within double quotes, the value of each parameter expands to a separate word.

$# The number of positional parameters in decimal.

$? The exit status of the last foreground command (i.e. the exit status of the last synchronous process or pipeline).

$$ The process ID, in decimal, of the invoked shell.

$! The process ID of the last background command.

$0 The name of the shell script, or the Shell itself.

Named parameters

Named parameters are values represented by a *name* as defined below. The named parameter is simply expanded into its corresponding value.

Note that if a named parameter is exported (via the **export** builtin), it becomes an environment variable.

The simplest form of named parameter expansion is:

$*name*

or

${*name***}**

where the value, if any, of *name* is substituted. Here is a simple example:

```
$ me=$LOGNAME
$ echo $me
bruce
```

A parameter expansion can be modified by one of the following methods:

${*parameter***:-***word***}**

If *parameter* is unset or null, *word* is substituted. Otherwise, the value of *parameter* is used.

${*parameter***:=***word***}**

If *parameter* is unset or null, *word* is assigned to *parameter*. Otherwise, the value of *parameter* is used as is. (Note that positional parameters and special parameters cannot be assigned in this way.)

${*parameter***:?**[*word*]**}**

If *parameter* is unset or null, *word* — or an error message from the Shell if *word* is omitted — is output to standard error. Otherwise, the value of *parameter* is used.

${*parameter***:+***word***}**

If *parameter* is unset or null, the null string is substituted; otherwise *word* is used.

In the parameter expansions shown above, using the colon (**:**) results in a test for the parameter being unset or null. If you omit the colon, the test is only for the parameter being unset.

Here are some examples of parameter expansion modified by the above methods:

```
$ unset handle
$ echo ${handle:-"handle is not set or null"}
handle is not set or null
$ echo ${handle-"handle is not set"}
handle is not set
$ echo ${handle:?}
missing value for handle
$ echo ${handle:?"handle is not set or null"} 2>error
$ cat error
handle is not set or null
$ echo $handle

$ handle=cup
$ echo ${handle-"handle is not set"}
cup
$ echo ${handle+"handle is set to something"}
```

© 2005, QNX Software Systems

```
handle is set to something
$ handle=""
$ echo ${handle-"handle is not set"}

$ echo ${handle:-"handle is not set or null"}
handle is not set or null
$ echo ${handle:="cup"}
cup
$ echo $handle
cup
```

In the parameter expansions below, *word* can contain patterns specified with **?**, *****, and **[**...**]** (these characters are described under "Filename expansion"):

${#*parameter***}**

The number of characters in the value of *parameter* is substituted.

${*parameter***%***word***}**

Strip the smallest suffix matching *word* from *parameter* before substituting *parameter*.

${*parameter***%%***word***}**

Strip the largest suffix matching *word* from *parameter* before substituting *parameter*.

${*parameter***#***word***}**

Strip the smallest prefix matching *word* from *parameter* before substituting *parameter*.

${*parameter***##***word***}**

Strip the largest prefix matching *word* from *parameter* before substituting *parameter*.

Here are some examples of parameter expansion modified by the above methods:

```
$ handle="cup of coffee"
$ echo ${#handle}
13
$ echo ${handle%"ffee"}la
cup of cola
$ echo ${handle}
cup of coffee
$ handle=foo.c
$ echo ${handle%.c}.o
foo.o
$ handle=/usr/local/bin/pcal
$ echo ${handle##*/}
pcal
$ handle=//1/home//bruce/src
$ echo /${handle#//*/}
/home//bruce/src
```

Tilde expansion

A tilde prefix consists of a tilde character (**~**) at the beginning of a word, followed by all the characters preceding the first slash (**/**) in the word, or all the characters if there is no slash. The tilde prefix is treated as a possible login name and is expanded into the pathname of the home directory associated with that login name. If the login name is null (i.e. the tilde prefix contains only the tilde character), the tilde prefix is expanded into the value of the **HOME** environment variable.

For example, if your home directory is **/home/fred**, the following lines are equivalent:

```
cat /home/fred/myfile

cat ~/myfile
```

© 2005, QNX Software Systems

Expressions

You can use expressions with the **let** command to assign a value to a variable; you can also use expressions as numeric arguments to the **test** command or as the value of an assignment to an integer variable.

Expressions may contain alphanumeric variable identifiers and integer constants. You can also combine them with the following operators (listed in decreasing order of precedence):

Unary operators

Operator:	Description:
-	unary minus
!	logical not

Binary arithmetic operators

Operator:	Description
*****	multiply
/	divide
%	modulus
+	add
-	subtract

Binary relational operators

Operator:	Description
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Equality operators

Operator:	Description
==	equal to
!=	not equal to

Precedence

You can override the precedence of the above operators with parentheses: **(** and **)**

Examples

Assign the value 3 to the variable *a*:

```
let a=3
```

Increment *a* by 3

```
let a=a+3
```

Assign the product of the variables *b* and *c* to *a*:

```
let a=b*c
```

Negate the value of *a*:

```
let a=-a
```

Create an integer variable *a*

```
typeset -i a
```

Assign the result of *b* modulo 2 to *a*:

```
a=b%2
```

The following example is a function to test if an integer is even (note the use of the arithmetic expression in the **test**):

```
function iseven
{
if test 0 -eq "($1 % 2)"
then
    echo $1 is even
else
    echo $1 is not even
fi
}
```

Note that the "return" value from this function is always true since, if a **return** *n* statement is not used to return from the function, the return value for the last command run in a function is the return value of the function.

Input line editing

When the **emacs** option is set (**set -o emacs**), the Shell provides interactive line editing. If the Shell is running on a tty, **emacs** line editing is enabled automatically. Note that the **vi** editing mode is not supported in the QNX shell.

In this line editing mode, which is similar to the emacs mode in AT&T's KornShell, various editing commands cause immediate actions without waiting for a **newline**. Several of these commands are bound to particular control characters when the Shell is invoked;

you can change those bindings with the **bind** builtin command. The default bindings resemble corresponding emacs key bindings.

Note that many of the editing commands are useful only on terminals with a visible cursor. In addition, any editing commands that refer to "hashing" will work only if hashing is enabled (**set -o hashall**).

Editing commands

In the list of commands below, the name of each command is followed by the command's default binding, if any. Most bindings are preceded by either of the following:

- **^** (caret) This indicates that you should press Ctrl at the same time as indicated character.
- **^[** This indicates that you should press Esc, then press the indicated character.

For example, the **^G** binding for the abort command indicates that you should press Ctrl – G.

abort

^G (Ctrl – G)
Aborts the search if given as a response to a request for a search-history pattern.

auto-insert

Causes the character to appear as literal input. Most ordinary characters (i.e. **a**, **b**, **c**...) are bound to this command.

backward-char

^B (Ctrl – B)
Moves the cursor back one character.

backward-word

^[b (Esc – B)
Moves the cursor to the beginning of a word. Words are delimited by the current setting of the **IFS** environment variable.

beginning-of-line

^A (Ctrl – A)
Moves the cursor to the beginning of the input line.

complete

^[^[(Esc Esc)
Automatically completes as much as is unique of the hashed command name or filename that has been entered.

The filename is determined by the partial path that has been entered (i.e. **/bin/cront** would yield **/bin/crontab**). If the filename doesn't begin with a **/**, the current directory is searched.

If the entire remaining command or filename is unique, a space is printed after its completion-unless it's a directory name, in which case **/** is appended.

If there's no hashed command or filename with the current partial word as its prefix, a bell character is output.

complete-command

^X^[(Ctrl – X Esc)
Completes as much as is unique of the hashed command name that has the partial word up to the cursor as its prefix. This is handy when substituting a command in the middle of a line.

Only command and function names seen since the last **hash -r** command are available for completion; you can use **hash** to register additional names.

complete-file

^X^X (Ctrl – X Ctrl – X)
Completes as much as is unique of the filename that has the partial path up to the cursor as its prefix.

delete-char-backward

^H (Ctrl – H)
Deletes the character before the cursor.

delete-char-forward

^D (Ctrl – D)
Deletes the character after the cursor.

delete-word-backward

^W (Ctrl – W)
Deletes characters before the cursor, back to the beginning of a word.

delete-word-forward

^[d (Esc D)
Deletes characters after the cursor, up to the end of a word.

down-history

^N (Ctrl – N)
Scrolls the history buffer forward one line (later). Each input line originally starts just after the last entry in the history buffer, so this command isn't useful until you've performed either search-history or up-history.

end-of-line

^E (Ctrl – E)
Moves the cursor to the end of the input line.

forward-char

^**F** (Ctrl – F)
Moves the cursor forward one position.

forward-word

^**[f** (Esc F)
Moves the cursor forward to the end of a word.

kill-line

^**U** (Ctrl – U)
Deletes the entire input line.

kill-to-eol

^**K** (Ctrl – K)
Deletes the input from the cursor to the end of the line.

list

^**[?** (Esc ?)
Prints a sorted, columnated list of hashed command names or filenames, if any, that can complete the partial word containing the cursor. Directory names have **/** appended to them, and executable file names are followed by *****.

This is useful if you want to find out what the possible matches might be for a "complete" editing sequence (e.g. ^**[**^**[**).

list-command

^**X?** (Ctrl – X ?)
Prints a sorted, columnated list of hashed command names, if any, that can complete the partial word containing the cursor. **list-file**

list-file

(unbound)
Prints a sorted, columnated list of filenames, if any, that can complete the partial word containing the cursor. Filetype indicators are

appended as described for “list,” above. By default, this command isn’t bound to a key sequence.

newline

^**J** or ^**M** (Ctrl – J or Ctrl – M)
Causes the current input line to be processed by the Shell. The cursor may be anywhere on the line.

quote

^^ (Ctrl – ^)
The character entered after the command is taken literally rather than as an editing command.

redraw

^**L** (Ctrl – L)
Reprints the prompt string and the current input line (only if it’s not the first character typed on a new line).

search-character

^**]** (Ctrl –])
Takes the character entered after the command and searches forward in the current line for the next occurrence of that character.

search-history

^**R** (Ctrl – R)
Searches the internal history list backwards for commands matching the input. An initial ^ in the search string anchors the search to the beginning of a line.

Successive search-history commands continue searching backward to the next previous occurrence of the pattern. The history buffer retains only a finite number of lines; the oldest are discarded as necessary.

Pressing **Esc** will leave search mode.

transpose-chars

^T (Ctrl – T)
Exchanges the character at the cursor with the previous character.

up-history

^P (Ctrl – P)
Scrolls the history buffer back one line (earlier).

yank

^Y (Ctrl – Y)
Inserts the most recently killed text string at the current cursor position (i.e. the string most recently killed by a KILL character such as **^U**).

yank-pop

^[y (Esc Y)
Immediately after a yank, replaces the inserted text string with the next most recently killed text string.

Initialization files

The Shell supports initialization files that allow both system-wide and user-specific customization. In these files, you can define aliases and functions as well as initialize environment variables.

/etc/profile

This is a system startup profile. It is the first file executed by every login shell. Local conventions (e.g. default paths, message-of-the-day) are often established through this file.

$HOME/.profile

This is a user-specific startup profile. It is executed by every login shell after the **/etc/profile** script. Exported environment variables are often set up in this file.

$ENV Every time a new shell is started, the new shell looks to see if the **$ENV** environment variable is defined. If it is, the

file that it specifies will be executed. This lets you set your aliases and functions (which are not inherited) so that they remain in your environment when you escape into a new shell.

To enable this feature, you must define and export **ENV**. This is typically done in your **.profile** file, as shown in this example:

```
export ENV=$HOME/.kshrc
```

Remember that every time a utility creates a shell on your behalf, this file will be executed. If many commands are placed in this file, you may notice some delay when these shells are started.

Pipes

A *pipe* is a mechanism built into the QNX filesystem that lets you connect the output of one program to the input of another without having to resort to a clumsy temporary file. A *pipeline* is a connection of two or more programs through pipes. The special character **|** informs the shell that a pipe is to be set up.

In the following example, the output of **ls** is piped to **wc** in order to count the number of lines in a file listing:

```
$ ls |wc -l
```

You can have as many programs in a pipeline as you like, as the following examples illustrate.

Count the number of times a particular user is logged on to this node:

```
$ who |grep steve |wc -l
```

Count the number of times that same user is logged on anywhere on the network. Also, save the output of the **who -a** command in the file **/tmp/whonet**:

```
$ who -a |tee /tmp/whonet |grep steve |wc -l
```

Count the number of nodes on the network and save the output of **sin -h net** in the file **/tmp/netinfo**:

```
$ sin -h net | tee /tmp/netinfo | \
  sed -e '/-----/,$d' | wc -l
```

Quoting

You use quoting to suppress the special meaning of certain characters and words to the Shell. You can use quoting to:

- preserve the literal meaning of certain characters
- prevent reserved words from being recognized as such
- prevent parameter expansion and command substitution within "here-document" processing

To quote a single character, sequences of characters, or words, you enclose them in single or double quotes. If only a single character needs to be quoted, it can be preceded with the escape character (\).

You *must* quote the following characters if you want them to represent themselves:

```
|  &  `  '  ;  \  (  )  "
```

plus spaces, tabs, and newlines.

You *might* need to quote the following characters, depending on their context within a shell command:

```
*  ?  [  #  ~  =  %
```

Escape character (backslash)

If not quoted, a backslash character (\) preserves the literal value of the next character, unless the next character is a **newline**. If a **newline** follows the backslash, the Shell interprets this as *line continuation*.

Single quotes

Enclosing characters and words in single quotes (**' '**) preserves the literal meaning of each character and reserved word within the single quotes. A single quote cannot be included with single quotes. For example:

```
echo 'What'\''s up Doc?'
```

results in:

```
What's up Doc?
```

Double quotes

Enclosing characters and words in double quotes (**" "**) preserves the literal meaning of all characters and words within the double quotes, with the exception of the **$**, ‘, and \ characters:

This character:	Retains its special meaning:
$	of introducing parameter expansion, command substitution, and arithmetic expansion
‘	of introducing the alternate form of command substitution (see “Command substitution”)
\	as the escape character

To include a double quote character within double quotes, you precede it with the escape character. For example:

```
toon=Bugs
echo "$toon: \"What's up Doc?\""
```

results in:

```
Bugs: "What's up Doc?"
```

Shell prompts

The Shell always prompts for input. By default, for non-**root** users, it displays this prompt:

```
$
```

This prompt is displayed whenever the Shell is waiting for input at the command line. The Shell also has a secondary prompt, which by default is set to:

```
>
```

The Shell displays this secondary prompt whenever a **newline** is entered and the Shell requires further input to complete the command (unless the **newline** has been escaped).

In the following example, the Shell presented the secondary prompt because the string wasn't terminated with a closing quotation mark.

```
$ echo "QNX 4 is POSIX
> and a whole lot more."
```

Redirection

When a command is run, part of its environment consists of standard files set up for its use. These files, standard input, standard output, and standard error output (*stdin*, *stdout*, *stderr*), are usually attached to the active terminal. When the Shell executes a command, the

command inherits a copy of the Shell's file descriptors without having to open them again. Before a command is executed, its input and output can be redirected using a special notation interpreted by the Shell. The general syntax used for redirection is:

[*n*]*redir-op file*

where *n* is an optional file descriptor number, *redir-op* is one of the redirection operators described below, and *file* is usually the name of a file.

Redirecting input

[*n*]**<***file* Opens *file* for reading on the specified file descriptor (*n*) or on standard input if the file descriptor isn't given.

Redirecting output

[*n*]**>***file* Opens *file* for writing on the specified file descriptor (*n*) or on standard output if the file descriptor isn't given. If the file exists, it's overwritten. If the file does *not* exist, it's created. If the **noclobber** option is set (see the **set** command) and the file already exists, [*n*]**>***file* will fail.

Redirecting output, ignore **noclobber**

[*n*]**>|***file* Identical to [*n*]**>***file* except that [*n*]**>|***file* will overwrite existing files even if the **noclobber** option is set.

Appending output

[*n*]**>>***file* Opens *file* for writing on the specified file descriptor (*n*) or on standard output if the file descriptor isn't given. If the file exists, output is appended to the end of the file. If the file does *not* exist, it's created.

© 2005, QNX Software Systems

The here-document

[*n*]**<<**[**-**]*word*

Reads the specified file descriptor *n*, or the standard input if a file descriptor isn't given, until a line that is the same as *word* is found or the end-of-file is reached. The resulting document becomes the standard input for the next command to be executed, hence the term "here-document."

If no characters in *word* are quoted, all lines of the here-document have parameter expansion, arithmetic expansion, and command substitution performed on them. In this case, you must use the backslash character (\) to quote these characters:

****, **$**, or ‘

If the form [*n*]**<<-***word* is used, leading tabs are stripped from the input lines and from the line containing the delimiter *word*. The following example will output the text between the first line, which contains the here-document operator (**<<**), and the last line, which consists of the delimiter (**THE_END**). The text between these two lines will be output without the leading tabs. This example could be used instead of a sequence of **echo** commands.

```
cat <<-THE_END
The QNX Operating System is ideal for
realtime applications. It provides
multitasking, priority-driven preemptive
scheduling, and fast context switching--all
essential ingredients of a realtime system.
THE_END
```

Duplicating an input or output file descriptor

[*n*]**<&***digit*

[*n*]**>&***digit*

The specified file descriptor *n*, or the standard input if *n* isn't given, is made to be a copy of the input file

descriptor specified by *digit*. The second form [*n*]**>&***digit* descriptor behaves similarly for duplicating an output file descriptor.

Opening a file descriptor for reading and writing

[*n*]**<>***file* Opens *file* for both reading and writing on the specified file descriptor or on standard input if the file descriptor isn't given.

Closing a file descriptor

[*n*]**<&-**
[*n*]**>&-** Closes the specified file descriptor *n*. If a file descriptor isn't given, the standard input is closed. The second form [*n*]**>&-** behaves similarly for closing an output file descriptor.

Remote execution

The preferred method of running a program remotely is to use the **on -n** *node* utility. However, the QNX shell does, for the time being, support a builtin syntax for executing programs on remote nodes. This feature is an extension to the shell and won't be found in other systems.

This syntax can be in one of two forms:

onnode *node command* [*argument*...]

or

//*node command* [*argument*...]

Both these forms of builtin shell support for remote execution are deprecated. Relying on them will reduce the portability of your code to future versions of QNX, and restrict your choice of shells which you can use with your QNX system to those supplied with the system. Use the **on** utility instead.

Remote execution is sometimes necessary when a utility has no way of performing the work on a remote system directly. An example of this is the **rtc hw** command, which needs access to the local hardware to fetch the current date and time from the clock hardware.

For example, the preferred way to load the OS clock on node 61 with the date and time from the hardware clock on the same machine is:

```
# on -n61 rtc hw
```

but the same can be accomplished (in a less portable fashion) using:

```
# onnode 61 rtc hw
```

or, even less desirably:

```
# //61 rtc hw
```

Reserved words

Reserved words are words that have special meaning to the Shell. The following words are reserved:

!	**elif**	**in**
{	**else**	**then**
}	**esac**	**until**
case	**fi**	**while**
do	**for**	
done	**if**	

Shell scripts

A *shell script* is a text file that contains a sequence of shell commands. A shell script may be executed in three ways

- from within an interactive shell as a command
- in batch mode as a command-line argument to the Shell (for more information, see the entry on "Invocation").
- if you have execute permission for it (see **chmod**), and it has the correct **#!/bin/sh** header (see discussion below) as the first line, it may be *spawn()*ed or *exec()*ed directly from any program.

The example on the following page is an implementation of the **which** utility as a QNX shell script. The example script is called **cpath**.

Of special interest within this shell script is the first line:

```
#! /bin/sh
```

This is a special syntax that informs the invoking shell, or the Process Manager (**Proc**) to spawn another shell (in this case, **/bin/sh**) to execute the shell script. For example, let's say the user's current shell is the Korn shell (**ksh**) and this shell script has been invoked by entering **cpath ls** at the command line. The Korn shell (or **Proc**'s loader thread) first reads this line, then invokes the QNX shell (**/bin/sh**) to execute the shell script.

If a script is being run from a shell and doesn't begin with **#!**, then **/bin/sh** is used anyway. If the script is being spawned directly (through *spawn*()*, qnx_spawn()* or *exec()* functions), then an exec format error will result if the script does not being with **#!** *programname*.

© 2005, QNX Software Systems

You'll find several more example shell scripts at the end of this man page.

```
#! /bin/sh

# Usage message compatible with the "use" utility starts below.

#ifdef __USAGE
#%C    - display which command in PATH is executed
#
#%C    command
#Where:
#    command     is the name of the command
#endif


oldpath=$PATH
PATH=/bin:/usr/bin

case $# in
0)      echo 'cpath command' >&2; exit 2
esac

for path in `echo $oldpath | sed 's/^:/.:/
                                 s/::/:.:/g
                                 s/:$/:./
                                 s/:/ /g'`
do
    if test -f $path/$1  # Is the command here?
    then
        echo $path/$1    # Yes. Display it and exit.
        exit 0
    fi
done

exit 1                   # Didn't find it...
```

Signals

When a command is an asynchronous command list (i.e. it is run in the background using the control operator **&**), the Shell prevents the SIGQUIT and SIGINT signals from interrupting the command. In the case of foreground commands, the Shell allows the same signals as its parent. Signals can also be *trapped* using the special builtin command **trap**.

Using Shell commands

A shell command is one of the following:

- control construct
- regular builtin utility
- special builtin command
- executable utility
- shell script

Commands invoked within the Shell have the following precedence:

1. special builtin commands
2. functions
3. regular builtin utilities
4. executable utilities and shell scripts

For example, when you enter:

```
umask
```

the Shell will execute its regular builtin **umask** instead of the executable utility **umask** since regular builtins have higher precedence than executable utilities.

The Shell special builtins, functions, and regular builtins are all resolved and executed within the Shell process. Executable programs and shell scripts are found using the **PATH** environment variable and are spawned as subprocesses.

Control Constructs

© 2005, QNX Software Systems

The `case` conditional construct

```
case word in
    pattern1)               list ;;
    pattern2|pattern3)      list ;;
    ...
esac
```

The **case** command executes the *list* associated with the first *pattern* that matches *word*. Patterns are the same as those used for filename generation except that the rules concerning the slash character and a leading dot don't apply.

The `if` conditional construct

```
    if expr
    then
        list
[
    elif expr
    then
        list
]...
[
    else
        list
]
    fi
```

The **if** *expr* is executed; if its exit status is zero, the **then** *list* is executed and the **if** construct is complete.

If the exit status of **if** *expr* is nonzero, however, each **elif** *expr* is executed in turn. If the exit status of an **elif** *expr* is zero, its corresponding **then** *list* is executed.

If the exit status of an **elif** *expr* is nonzero, the **else** *list*, if any, is executed.

The `for` loop

```
for name [ in word... ]
do
    list
done
```

The variable *name* is set to each *word*, in turn, and *list* is executed each time. If the **in** *word*... is omitted, then the **for** command executes *list* once for each positional parameter set (i.e. the equivalent of **for** *name* **in "$@"**)

The `while` loop

```
while list1
do
    list2
done
```

The **while** command executes *list1*; if *list1*'s exit status is zero, *list2* is executed and the loop is repeated. Otherwise, the loop is exited.

The `until` loop

```
until list1
do
    list2
done
```

The **until** command executes *list1*; if *list1*'s exit status is nonzero, *list2* is executed and the loop is repeated. Otherwise, the loop is exited.

The `function` definition command

```
name ()
{
    list
}
```

or

```
function name
{
    list
}
```

Define a function *name* that is used by the Shell as a simple command. Any operands passed to the function can be accessed within the function using the positional parameters. The special parameter **$#** is also temporarily overridden to be the number of operands passed to the function. The body of the function is the *list* of commands.

Here's an example shell function that would do context-sensitive **grep**s. You can pass it any number of parameters. The string is assumed to be the first argument and the rest are shifted through and **grep**'d:

```
cgrep () {
   str=$1; shift
   for f in $@; do
       grep -v $str $f | diff -c - $f  #context grep
   done
}
```

Regular builtin utilities

These "regular" utilities may be found external to the Shell, but they have been built into the Shell to enhance system performance.

The following regular builtin utilities are briefly described here. For a more complete description of **cd**, **false**, **kill**, and **true**, see their respective man pages.

cd

Change working directory.

cd [*directory*]

The **cd** utility changes the working directory of the current execution environment. If *directory* isn't specified, the value of the **HOME** environment variable becomes the new working directory.

false

Return a false value.

false

The **false** utility returns a nonzero exit status.

getopts

Parse utility options.

getopts *optstring name* [*argument*...]

The **getopts** utility uses the characters in *optstring* to parse and retrieve options and option arguments from a list of parameters. The *optstring* argument is a string containing the option characters to be recognized. A colon following an option character informs **getopts** that this option is expected to have an option argument.

By default, **getopts** parses the positional parameters passed to the invoking shell procedure. If one or more arguments are given, the arguments are parsed instead of the positional parameters.

Each time it's invoked, the **getopts** utility places the value of the next option in the shell variable specified by *name*; it also places the index of the next argument to be processed in the shell variable *OPTIND*. When the Shell is first invoked, *OPTIND* is initialized to 1. When an option requires an option argument, it is stored in the variable *OPTARG*. If no option is found, or if an option is found but it doesn't have an option argument, *OPTARG* is unset.

If an option character not contained in *optstring* is found, *name* is set to the question mark character (**?**), *OPTARG* is unset, and an error message is written to the standard error output. Under this condition, if the first character of *optstring* is a colon, *OPTARG* is instead set to the option character found and no error message is output.

When the **getopts** utility has processed all of the options, it returns with an exit status greater than zero and the **OPTIND** variable is set to the index of the first nonoption argument.

history

Display an entry or range of entries from the history list.

history [*start_number* [*end_number*]]

The Shell keeps a list of the text strings that have been submitted for execution — this is known as the Shell's *history*. You can quickly recall and execute entries from your history (e.g. via Ctrl – P or Ctrl – R). For more information, see the entry on "Input line editing."

When invoked without arguments, **history** displays the last 13 entries in the history list.

kill

Terminate or signal processes.

kill *signame pid*...

kill -*signum pid*...

kill -l

The **kill** utility sends the signal specified by *signame* or *signum* to the specified processes (if a *pid* is negative, it specifies a process group).

In the first form, *signame* is the symbolic name of a signal. Values of *signame* are recognized in case-independent fashion, with or without the **SIG** prefix. In the second form, *signum* is the decimal number of the signal to be set. If no signal is specified, the termination signal is set by default.

If the **-l** (el) option is specified, the **kill** utility writes a list of all supported signal names to the standard output.

onnode

Start a process on another node.

onnode *node command* [*argument*...]

The **onnode** builtin is a deprecated shell feature which is a quick way of doing the equivalent of what **on -n** *node* does. It is used for remote execution. It executes the command with the specified arguments on the specified node. The standard input, standard output, and standard error output are inherited from the invoking shell. The syntax **onnode 1 sin** is the same as:

```
//1 sin
```

which is similarly discouraged.

Instead of these builtins, you should use the **on** utility to run a command on another node.

ontty

Start a process on another tty.

ontty *device command* [*argument*...]

The **ontty** builtin is a deprecated shell feature which is a quick way of doing the equivalent of what **on -t** *tty* does. It executes the command with the specified arguments on the named tty. The command is detached from the invoking terminal. The **ontty** builtin operates by redirecting the program's *stdin*, *stdout*, and *stderr* to the indicated device. If the device is remote, the **ontty** command remotely executes the program on that node.

In the following example, **mymonitor** is run on the local node, with its input and output attached to **/dev/ser1** of node 4:

```
ontty //4/dev/ser1 mymonitor
```

In the next example, **monitor** runs on node 4 while its I/O is attached to **/dev/ser1** on the local node:

```
onnode 4 ontty /dev/ser1 monitor
```

Instead of this builtin, you should use **on** utility to run a command on another tty.

© 2005, QNX Software Systems

read

Read a line from standard input.

read [**-r**] *var*... [**?** **"***prompt_string***"**]

The **read** utility reads a single line from the standard input. The line is split into words using the separator characters contained in the *IFS* variable. The first word is assigned to the first *var*, the second word to the second *var*, and so forth. Any additional words are appended to the last *var*. If the number of *var* arguments specified is greater than the number of words in the input line, the extra *var* arguments are assigned the null string.

If the **-r** option is specified, the backslash character (\) is treated like any other character in the input line instead of being interpreted as an escape character.

If a question mark (**?**) follows the variables, the shell will print the text following the question mark as a prompt before reading the input. For example, to read data into a variable **$a**, one might use:

```
$ read a?"Enter value for a:"
Enter value for a: blahblah
$ echo $a
blahblah
```

test

Test condition.

test *expression* [*expression*]

The **test** utility evaluates the expression and returns zero status if true and nonzero status otherwise. It is normally used as the controlling command of the **if** and **while** statements.

The following basic expressions are available:

Expression	Meaning
-r *file*	*file* exists and is readable
-w *file*	*file* exists and is writable
-x *file*	*file* exists and is executable
-f *file*	*file* is a regular file
-d *file*	*file* is a directory
-c *file*	*file* is a character special device
-b *file*	*file* is a block special device
-p *file*	*file* is a named pipe
-u *file*	*file* mode has setuid bit
-g *file*	*file* mode has setgid bit
-k *file*	*file* mode has sticky bit
-s *file*	*file* is not empty
-L *file*	*file* is a symbolic link
-S *file*	*file* is a socket
file1 **-nt** *file2*	*file1* is newer than *file2*
file1 **-ot** *file2*	*file1* is older than *file2*
file1 **-ef** *file2*	*file1* is the same file as *file2*
-t *filedes*	file descriptor is a tty device
string	*string* is not null
-z *string*	*string* is null
-n *string*	*string* is not null
*string1***=***string2*	*string1* and *string2* are equal
*string1***!=***string2*	*string1* and *string2* are not equal

continued...

Expression	Meaning
num1 **-eq** *num2*	*num1* and *num2* compare equal
num1 **-ne** *num2*	*num1* and *num2* compare not equal
num1 **-ge** *num2*	*num1* compares greater than or equal to *num2*
num1 **-gt** *num2*	*num1* compares greater than *num2*
num1 **-le** *num2*	*num1* compares less than or equal to *num2*
num1 **-lt** *num2*	*num1* compares less than *num2*

You can combine the above basic expressions with these operators:

Operator(s)	Example	Description
-o	*expr* **-o** *expr*	logical **OR**
-a	*expr* **-a** *expr*	logical **AND**
!	**!***expr*	logical **NOT**
(...**)**	**(***expr***)**	precedence grouping

true

Return true value.

true

The **true** utility returns a zero exit status.

wait

Await process termination.

wait [*jobid*...]

The **wait** builtin waits for the specified jobs to terminate and returns the exit status of the process requested by the last *jobid* operand. If no operands are specified, it waits for all of the child processes of the

invoking shell to terminate. The *jobid* may be specified in the following forms (the **%** character is optional):

%+	Select the current job
%-	Select the previous job
%*n*	Select job number *n* (see output of **jobs** builtin).
%*string*	Select job whose command starts with *string*.
%?*string*	Select job whose command contains *string*.

Special builtin commands

The following special commands are "built into" the Shell; that is, the Shell interprets and executes them internally. Because these commands are tightly bound to the shell interface, they are called *special builtin commands*. The output of these commands, if any, is written to the standard output; you can apply normal redirection and piping operations.

alias

List aliases.

alias [*name***=***value*...]

Without arguments, **alias** lists all aliases and their values. If a name is specified without a value, its value is listed. Any name specified with a value defines an alias. System V command hashing is implemented (see **hash**).

Alias expansion occurs when the first word of a statement is a defined alias, except when that alias is already being expanded. It also occurs after the expansion of an alias whose definition ends with a space.

List all aliases:

```
alias
```

Show the current definition of a specific alias:

alias *alias_name*

Remove an alias:

unalias *alias_name*

Some common aliases:

```
alias l="/bin/ls -lFR"
alias lf="/bin/ls -CF"
alias m=less
```

bind

Create bindings for commands or text strings.

bind

bind *string***=***editing_command*

bind -m *string***=***substitute*

Without arguments, the **bind** command lists the current bindings.

With the *string***=***editing_command* syntax, the specified editing command is bound to the given string. If the string is subsequently input, the editing command will be immediately invoked. The string, which should consist of a character, may be preceded by either of the following:

^	(caret) This indicates that Ctrl should be pressed at the same time as the string.
^[	This indicates that Esc should be pressed before the string.

For example, if you created the following binding for the **forward-char** command:

```
bind ^f=forward-char
```

You would press Ctrl – F to use the binding.

With the **-m** option, the given input string, whenever subsequently specified, will be immediately replaced by the given substitute string. The substitute string may contain editing commands.

Editing commands are described in the entry on "Input line editing."

break

Exit from a **for**, **while**, or **until** loop

break [*n*]

The **break** command exits from the smallest enclosing **for**, **while**, or **until** loop, or from the *n*th enclosing loop if *n* is specified (default for *n* is 1). If *n* is greater than the number of enclosing loops, then the largest enclosing loop is exited. Execution continues with the command immediately following the exited loop.

builtin

Execute a command as a builtin command.

builtin *command* [*argument*...]

The **builtin** command causes the Shell to treat the specified command as a builtin command, suppressing the shell function lookup that is part of the default command lookup process.

: (colon)

Null command.

:

The **:** (colon) command does nothing; it is simply a null command or place holder. It always returns an exit status of zero.

continue

Continue a **for**, **while**, or **until** loop.

continue [*n*]

The **continue** command returns to the top of the smallest enclosing **for**, **while**, or **until** loop, or to the top of the *n*th enclosing loop if *n* is specified (default for *n* is 1). If *n* is greater than the number of enclosing loops, then execution resumes at the top of the largest enclosing loop.

. (dot)

Execute commands in the current environment.

. *file*

The **.** (dot) command reads and executes the commands from *file* within the current environment. The search path contained in the environment variable **PATH** is used to find the directory containing *file*. This command doesn't require that *file* be executable.

eval

Execute a command from concatenated arguments.

eval [*argument*...]

The **eval** command constructs a command by concatenating the specified arguments. The constructed command is then read as input to the Shell and executed.

exec

Execute a command and/or manipulate file descriptors.

exec [*command* [*argument*...]]

The **exec** command opens, closes, and/or copies file descriptors as specified by any I/O redirections given as part of *argument*. If a command is specified, that command is spawned as a replacement for the Shell. Any specified arguments are passed to the spawned process.

exit

Cause the Shell to exit.

exit [*n*]

The **exit** command causes the Shell to exit with the exit status specified by *n*. If *n* isn't specified, the exit status is that of the last command executed.

export

Mark environment variables for export.

export *name*[**=***word*]...

export -p

The **export** command marks the specified names for **export**, which will cause them to be in the environment of subsequently executed commands. If you specify the **-p** option, the names and values of all exported variables are written to the standard output.

hash

Print hashed command pathnames.

hash [**-r**] [*name*...]

Without arguments, **hash** lists any hashed executable command pathnames. If any *name*s are given, each is searched as if it were a command name and is added to the hash table if it is an executable command. The **-r** option removes all commands from the hash table.

Hashing is enabled by the **set** builtin command (**set -o hashall**). When hashing is enabled, the location of each executed command is memorized (i.e. hashed). In addition, the hash table is searched in preference to the **PATH** when any command is invoked.

jobs

Display status of jobs in the current session.

The **jobs** command displays the status of jobs that were started under the current shell.

let

Evaluate expressions.

let [*argument*...]

Without arguments, **let** returns a nonzero exit status. If arguments are included, each expression is evaluated (see the "Expressions" entry). A zero status is returned if the last expression evaluates to a nonzero value; otherwise a nonzero status is returned.

print

Print arguments.

print [*argument*...]

The **print** command prints its arguments on the standard output, separated by spaces, and terminated with a **newline**.

readonly

Mark environment variables as read-only.

readonly *name*[**=***word*]...

readonly -p

The **readonly** command marks the specified names as "read-only." Environment variables marked with the *readonly* attribute cannot be changed by a subsequent assignment nor can they be unset by the **unset** command.

If the **-p** option is given, the names and values of all readonly variables are written to the standard output.

reopen

Attach input and output to a device. (*Deprecated*)

reopen *device*

The **reopen** command closes standard input, standard output, and standard error and attaches them to the specified device. This command is often used in **sysinit** files.

The following would reopen standard I/O to the new device **//0/dev/con1**:

```
reopen //0/dev/con1
```

This builtin is deprecated in favor of its non-QNX-specific equivalent,

```
exec <>//0/dev/con1 >&0 2>&0
```

return

Return from a function.

return [*n*]

The **return** command causes the Shell to exit the current function or **dot** script. The return value is either the value specified by *n* or the exit status of the last command executed if *n* isn't specified.

set

Set/unset options and positional parameters.

set [**-aefhkmnquvx**] [**-o** *option_name*]... [*argument*...]

set [**+aefhkmnquvx**] [**+o>** *option_name*]... [*argument*...]

set -- [*argument*...]

The **set** command sets and/or unsets the shell options. An option preceded by:

- **-** sets the corresponding attribute
- **+** unsets the corresponding attribute

The current set of option flags is kept in the special parameter **-**.

© 2005, QNX Software Systems

If any *argument*s are given, the positional parameters are overridden with these values and the special parameter **$***n* is updated. Note also that the double dash (**--**) is used to separate the options from the *argument*s.

set Option	**Description**
--	Unset all positional parameters. If any arguments are specified, define where they begin.
-a	Set the export attribute for each variable that is created or assigned a new value.
-e	Exit immediately if a command returns with a nonzero exit status.
-f	Don't expand filenames.
-h	Add command pathnames to the hash table
-k	Allow variable assignments to be recognized anywhere in the command.
-m	Monitor background jobs.
-n	Read commands but don't execute them. This option can be used to check a shell script for syntax errors before actually executing it. This option is ignored in interactive mode.
-o *option_name*	Set the specified option. (See entries below for details.)
-o allexport	(same as **-a**)
-o errexit	(same as **-e**)
-o keyword	(same as **-k**)

continued...

set Option	Description
-o monitor	(same as **-m**)
-o noexec	(same as **-n**)
-o noglob	(same as **-f**)
-o nounset	(same as **-u**)
-o verbose	(same as **-v**)
-o trackall	(same as **-h**)
-o xtrace	(same as **-x**)
-o bgnice	Run background jobs with lower priority.
-o emacs	Enable emacs-style line editing (default for ttys).
-o hashall	Enable command hashing.
-o ignoreeof	Don't exit on end-of-file (the Shell will exit only if you enter **exit**).
-o noclobber	Don't overwrite existing files. This prevents the Shell's **>** redirection operator from overwriting existing files. You can use the **>|** redirection operator to override this option for individual files. To display current settings for **-o**, type **set -o**.
-q	Ignore the QUIT signal.
-u	Treat the expansion of unset named parameters as an error.
-v	Be verbose; write input lines to standard error before execution.
-x	Enable execution trace; write each command and its arguments to the standard error output before execution.

times

Print system and user times.

times

The **times** command lists the system and user times for your invocation of the Shell. If any child processes created by the Shell have died, their times are also listed.

shift

Shift positional parameters.

shift [*n*]

The **shift** command shifts positional parameters down by *n* positions. If *n* isn't specified, it defaults to 1. Positional parameter 1 becomes parameter (1+*n*), parameter 2 becomes parameter (2+*n*), and so forth. The special parameter **$#** is updated to contain the new number of positional parameters.

typeset

Print variables and their attributes.

typeset [**+firx**] [*name*[**=***value*]...]

If no arguments are given, **typeset** lists all variables and their attributes.

If names are given, **typeset** sets the attributes of the named variables. Variables may also be assigned a value. If used inside a function, the created variables are local to the function.

If options but no names are given, **typeset** lists variables with specified attributes and their values (unless **+** is used).

The attributes are as follows:

-f List functions instead of variables.

-i Store the variable's value as an integer.

-r Make the variable read-only so that it cannot be reassigned a value.

-x Export the variable to the environment.

trap

Trap signals.

trap [*action condition*...]

The **trap** command causes the specified *action* to be executed when one of the specified *condition*s occurs. A *condition* is either a signal specified using a symbolic name, with or without the **SIG** prefix, or the string "**EXIT**". The *action* is simply a Shell command and is processed by the Shell in a manner equivalent to:

```
eval "$action"
```

Signals that were ignored on entry to a non-interactive shell cannot be trapped or reset, whereas an interactive shell may reset or catch them. Traps remain in place for a given shell until they are explicitly changed with another **trap** command.

If no arguments are given, the **trap** command writes a list of *action*s associated with each *condition* to the standard output.

unalias

Remove aliases.

unalias *name*...

The **unalias** command removes the aliases for each given name.

unset

Unset values and attributes of named parameters and functions.

unset [**-f**] *name*...

The **unset** command unsets each named parameter or function specified. If the **-f** option is specified, then the function *name* is

unset. If the **-f** option is *not* specified, then *name* refers to a named parameter. If that named parameter doesn't exist, then the function of that name, if any, is unset.

You can't use the **unset** command to unset read-only named parameters, positional parameters, and special parameters.

whence

Print command types.

whence [**-V**] *name...*

The **whence** command prints the type of command for each given name. If you specify **-V**, function and alias values are also printed.

Sample shell scripts

You can find the source for the following shell scripts in the **/usr/demo/scripts** directory.

lib2list — create sorted list of routine names

```
#! /bin/ksh
#set -vx

#
#    Use: lib2list <library_name>
#
#    Create a sorted list of routine names from a Watcom library
#
cd /usr/lib

if test $# -eq 0
then
    echo Use: ${0##*/} "<library_name>"
    exit 0
fi

tmp=$TMPDIR/l2l.$NODE.$$-

#  The "trap" routine to clean up if user hits ^C (intr)
trapper ()
{
    rm -f ${tmp}*
    unset tmp
    exit 1
}

trap trapper INT QUIT

#  Check if the library is accessible. Try the library name
#  as first given, then try it with an appended ".lib"
```

```
#  suffix, then try it with the "large model" character
#  inserted, then give up.

if test -f $1
then
    lib=$1
elif test -f $1.lib
then
    lib=$1.lib
elif test -f $1l.lib
then
    lib=$1l.lib
elif
then
    echo ${0##*/}: unable to open library \"$1\".
    exit 1
fi

echo Library name list for \"${lib}\":

#   Use "sed" to filter the list of names from the
#   "wlib" output. Then "cut" out the two columns into
#   a new single-column file.
#
wlib ${lib} |sed -e '/^WATCOM/,/^WATCOM/d' -e '/^$/,$d'\
-e 's/  */:/g' >${tmp}A
cut -d: -f1 ${tmp}A >${tmp}B
cut -d: -f2 ${tmp}A >>${tmp}B
#

#  Use the following "sed" line instead to see all the names.
#  (i.e. double and treble underscore names)
#
#sed -e '/^ *$/d' -e '1,$s/_*\.\.*.*$//' ${tmp}B |sort

sed -e '/^ *$/d' -e '1,$s/_*\.\.*.*$//' -e '/^__.*$/d' ${tmp}B\
|sort

#  Clean up the files and variables we used.
rm -f ${tmp}*
unset tmp

exit 0
```

`reverse` — look at options and process a string

```
#! /bin/sh
#ifdef __USAGE
#%C [options] your_word
#
#Options:
# -b      What the word begins with
# -c      Number of characters in the word
# -r      The word in reverse
#endif

#set -v
```

© 2005, QNX Software Systems

```
me=${0##*/}

print_usage ()
{
    echo "$me [options] your_word"
    echo
    echo "Options:"
    echo " -b      What the word begins with"
    echo " -c      Number of characters in the word"
    echo " -r      The word in reverse"
    exit 0
}

trapper ()
{
    echo Script interrupted by user!
    echo exiting...

    exit 0
}

#
#   Reverse the string passed in the positional parameters
#   and output the resulting string on the standard output
#   (file descriptor 1).
#
reverse ()
{
    typeset newword word x y
    typeset -i len

    newword=""
    word="$*"
    len=${#word}

    while test $len -gt 0
    do
        x=${word%?}
        y=${word##$x}
        word=$x
        newword=$newword$y
        let len=len-1
    done

    echo $newword
}

#
#   The main body of the shell script
#

trap trapper INT QUIT

if test $# -lt 1
then
    print_usage
fi

#   Check for options
#
```

```
typeset -i bflag=0 cflag=0 rflag=0

while getopts bcrx:y: opt $*
do
    case $opt in
         b)  bflag=1 ;;
        +b)  bflag=0 ;;
         c)  cflag=1 ;;
             r)  rflag=1 ;;
         x)  echo Argument to \"-x\" option is \"$OPTARG\"  ;;
         y)  echo Argument to \"-y\" option is \"$OPTARG\"  ;;
        \?)  echo ; echo ; print_usage ;;
    esac
done

let i=OPTIND-1

shift $i
word="$*"

if test $bflag -eq 0 -a $cflag -eq 0 -a $rflag -eq 0
then
    bflag=1;
    cflag=1;
    rflag=1;
fi

echo Your word \"$word\":

if test $bflag -eq 1; then
    case $word in
      [0123456789]*) echo "  begins with a number." ;;
      [a-zA-Z]*)     echo "  begins with a letter." ;;
      *)             echo "  starts with a special character." ;;
    esac
fi

if test $cflag -eq 1; then
    echo "    has ${#word} characters in it."
fi

if test $rflag -eq 1; then
    echo "    in reverse is \"`reverse $word`\"."
fi

unset me bflag cflag rflag

exit 0
```

Examples:

Invoke a subshell from the command line:

```
sh
```

© 2005, QNX Software Systems

Invoke a subshell with a command script:

```
sh /etc/backup
```

Invoke a subshell with a command string:

```
sh -c "cd /etc; ls -alF"
```

See also:

esh, **ksh**, **uesh**

show_pci

Display PCI devices (QNX)

Syntax:

```
show_pci [-v]
```

Options:

-v Verbose.

Description:

The **show_pci** command is used for diagnostic purposes to provide information on PCI Local Bus configuration. This information can be used to troubleshoot misconfigurations and to assist in finding out correct options to specify to device drivers.

Detailing the significance of all the information returned by **show_pci** is beyond the scope of this document. However, here are the meanings of some of the common fields returned by **show_pci**:

PCI version	This is the version of the PCI bus being used in the machine.
Vendor ID	A 16-bit number indicating the manufacturer of the device. Vendor IDs are allocated by the PCI Special Interest Group. The **show_pci** utility is aware of some of the more common ones. For those that it knows, it will display the manufacturer name after the hexadecimal number. (e.g. 8086h, INTEL CORPORATION)
Device ID	A 16-bit number identifying the particular device. The device ID is allocated by the device manufacturer (see Vendor ID).
PCI index	The PCI index corresponds to the *N*th occurrence of a device. For example if there are two identical devices in a machine the BIOS will assign them a PCI index of 0 and 1.

Class Code — The class code register identifies the general function of the device. The **show_pci** command will display the meaning of known classes after the hexadecimal class number. e.g. **10180h Mass Storage (IDE) ProgIF=128**

Revision ID — An 8-bit number which identifies a device-specific revision number, chosen by the device manufacturer (see Vendor ID).

Bus number — This is a number in the range 0 to 255 that uniquely selects a PCI bus.

Device number — This is a number in the range 0 to 31 that uniquely selects a device on a PCI bus.

Function num — This is a number in the range 0 to 7 that uniquely selects a function within a multi-function PCI device.

Status Reg — The contents (in hex) of the device's status register.

Bit	Meaning
0-4	Reserved.
5	66MHz capable (otherwise 33MHz)
6	Device supports UDF (user-definable features).
7	Capable of accepting fast back-to-back transactions when the transactions are not to the same agent.
8	(bus masters only) Data parity error detected.

continued...

Bit	Meaning
9-10	**DEVSEL#** encoding. 00=fast, 01=medium, 10=slow
11	Signaled Target Abort. This bit must be set by a target when it terminates a transaction with Target-Abort.
12	Received Target Abort. This bit must be set by a master device when its transaction is terminated by a Target-Abort.
13	Received Master Abort. This bit must be set by a master device whenever its transaction (except for Special Cycle) is terminated with a Master-Abort.
14	Signaled System Error. This bit is set whenever the device asserts **SERR#**.
15	Detected Parity Error. This bit is set when the device detects a parity error, even if parity error handling is disabled.

Command Reg The contents (in hex) of the device's command register.

Bit	Meaning
0	IO Space. When set, allows the device to respond to I/O Space accesses.
1	Memory Space. When set, allows the device to respond to Memory Space accesses.
2	Bus Master. When set allows the device to behave as a bus master.

continued...

Bit	Meaning
3	Special Cycles. When set, allows the device to monitor special cycle operations.
4	Memory Write and Invalidate Enable. When set, masters may generate the Memory Write and Invalidate command. When not set, the Memory Write command must be used instead.

Header type — This byte identifies the layout of the device's configuration space header and also indicates whether or not the device contains multiple functions.

Bit	Meaning
0-6	These 7 bits identify the header layout. The encoding 00h is the standard device header, while the encoding 01h specifies the header layout used for PCI-to-PCI bridges.
7	If set, this indicates a multi-function devices. Otherwise, the device is a single-function device.

BIST — Built-in Self Test control and status (9 bit register). Devices that do not support BIST will always return 0.

Bit	Meaning
7	BIST Capable. If set, the device supports BIST.

continued...

Bit	Meaning
6	Start BIST. Software invoking BIST will write a 1 to this bit location. The device will reset to zero when the BIST is complete.
5-4	Reserved. Always 0.
3-0	Completion code. A value of 0 means the device passed the test. Non-zero values indicate failure. Device-specific failure codes can be encoded into the available 4 bits.

Latency Timer — This 8-bit value specifies (in units of PCI bus clocks) the value of the latency timer for a PCI bus master device.

Cache Line Size — This 8-bit register specifies the system cacheline size (in units of 32-bit words).

Max Lat — This value specifies the latency that the device will tolerate in obtaining access to the PCI bus. The value is reported by **show_pci** in nanoseconds.

Min Gnt — This value specifies how long a burst period the device needs assuming a clock rate of 33MHz. This value is reported in nanoseconds by the **show_pci** utility.

PCI Int Pin — This register specifies which interrupt pin the device uses:

Value	Meaning
1	**INTA#**
2	**INTB#**
3	**INTC#**

continued...

Value	Meaning
4	**INTD#**

Devices that do not use an interrupt pin will always put 0 in this register.

Interrupt Line — This is an 8-bit value used for interrupt line routing information. The value in this register indicates which input of the system interrupt controller(s) the device's interrupt pin is connected to. The device itself does not use this value.

Examples:

Show devices on the PCI bus:

```
show_pci
```

See also:

PCI Local Bus Specification, PCI Special Interest Group, P.O. Box 14070 Portland, Oregon, USA 97214.

shutdown

Shut down and reboot the system (QNX)

Syntax:

shutdown [-f] [-n *node***]**

Options:

-f	Shut down fast. Send a SIGPWR signal, but wait only one second before rebooting.
-n *node*	Shut down the specified node (default is current node).

Description:

The **shutdown** utility performs an orderly system shutdown. This consists of first causing a SIGPWR to be sent to all processes, waiting for ten seconds, shutting down the filesystem, then performing a reboot of the system. The ten-second interval allows processes that have elected to catch the SIGPWR signal to perform any cleanup they need to do before the system is rebooted.

The **shutdown** utility sends messages to the **Proc** manager to accomplish both the sending of SIGPWR to all processes and the rebooting of the system. Some PC motherboards have problems with the standard methods of performing soft-reboots. If the system hangs after the ten-second countdown instead of rebooting, look at the options to the **Proc** manager to see if your version of **Proc** has an option to use an alternate method of soft-rebooting the system (if so you would have to add this to the invocation of **Proc** in the OS build-file – see **buildqnx**).

Recent versions of **shutdown** (QNX 4.25 and later) send a SIGTERM to **Fsys** after the SIGPWR and ten-second wait. After issuing the SIGTERM, **shutdown** waits for **Fsys** to terminate before proceeding to either reboot or indicate that it is safe to reboot. Early versions of the filesystem would terminate immediately upon receipt of SIGTERM, while more recent versions (QNX 4.25 and later) will perform an orderly shutdown of the filesystem. In 99% of the cases, the filesystem will already be idle when the SIGTERM is issued. However, using up-to-date versions of **shutdown** and **Fsys** will

© 2005, QNX Software Systems

provide greater assurance that the filesystem was indeed idle before the system was rebooted.

You must be **root** to use **shutdown**.

See also:

Proc

Installation & Configuration

Syntax:

Obtain information about processes:

sin [*options***]**

sin [*options***] args|dir|env|family|flags|memory|**
registers|root|rtimers|signals|
times|users

sin [*options***] format** *format_str*

sin [*options***] fds|files**

Obtain information about other OS resources:

sin [-n *node***] [-h] gdt|idt|irqs|names|**
gnames|proxies|sessions

sin [-n *node***] [-h] [-l] vcs**

sin [-n *node***] freemem|tree**

Obtain general system information:

sin [-n *node***] [-h] net|versions**

sin [-n *node***] info**

Options:

-g *groupname*	Get information on this group name only.
-h	Don't display headers. Works on all variations of the **sin** command except **sin freemem**, **sin tree** and **sin info**.
-l	Keep requests local. Has an effect only when used with **sin vcs**.
-m	Get information on processes in your own session only.

-n *node* Get information from this node. Works for all variations of the **sin** command.

-p *pid* Get information on this process ID only.

-P *string* Get information on processes whose names contain *string*.

-s *sid* Get information on this session ID only.

-S {U|b|g|i|n|p|s|t|u}
Sort on the specified *format_str* (see below). Only a single letter can be specified.

-u *username* Get information on only this user name.

Description:

The **sin** utility lets you obtain a "snapshot" of the state of the system (**s**ystem **in**formation). If present, the keyword after the command lets you select different types of information and different views of how the information is displayed. You need only type the first two characters of a keyword.

The information can be grouped into three classes as follows:

- Process information queries
- Other OS resource information queries
- General information

The **-n** *node* option may be used for any of the three classes of query to obtain information about another node on the network. The default is to obtain information on the current node.

The **-gmpPsu** options can be used only with the first class of queries (process information). Each option acts as a filter reducing the information displayed.

The **-l** ("el") option can be used only with the **vcs** query ("other OS resources"). It prevents **sin** from going off the current node to obtain the names of the processes on the remote end of a virtual circuit.

Process information queries

The first class gives reports on each process in the system. Since this information exceeds what can be displayed on a line, the **format** keyword allows you to select the components you wish displayed. A number of pre-specified reports have been provided to save typing in the various format options:

Command:	Description:
sin format *format_str*	Displays the requested info about each process.
sin	same as **sin format sinbpm**
sin args	same as **sin format iLa**
sin dir	same as **sin format iLnd**
sin env	same as **sin format iLne**
sin family	same as **sin format snpbif**
sin flags	same as **sin format sinbF**
sin registers	same as **sin format ir**
sin root	same as **sin format iLnR**
sin signals	same as **sin format sinpS**
sin times	same as **sin format sinpt**
sin users	same as **sin format signUu**

Output from the first class of queries is sorted by process ID. This can be changed by specifying the **-S** option with an alternate *format_str* (e.g. session ID, or perhaps priority).

format_str; Categories of Process Information

The *format_str* argument to **sin format** determines the categories of information that will be reported about each process. These categories are as follows:

© 2005, QNX Software Systems

Character	Represents
a	Command-line arguments
b	Process state/blocked on
d	Current Directory
e	Environment Variables
F	Process Flags
f	Family (Dad/Son/Brother relationships)
g	Process Group ID
i	Process ID
L	Login Name
M	Magic
m	Size
N	Session Node ID
n	Process Name
P	Memory Pass
p	Priority
R	Pathname Prefix Root
r	Registers
S	Signals
s	Session ID
T	File Time and Execution Start Time
t	CPU Time Used
U	Real User ID/Group ID
u	Effective User ID/Group ID

Command-Line Arguments

a The **a** format character causes an ARGUMENTS heading and the arguments of each process to be displayed. If you do not have appropriate permissions, a "Not available." message will be displayed under the ARGUMENTS heading. If the argument list exceeds 512 characters, **sin** will truncate the list and display a " ... " at the end to signify that truncation occurred. Unless you are **root** you can obtain command line arguments only for processes which are being run by your own user ID.

Process State/Blocked On

b The **b** (for "blocked") format character causes two headings to be displayed. STATE is used to display the current state of each process. The second heading, BLK, gives the PID that the process is blocked on (if any).

Possible values of STATE are:

State	Description
READY	this process is ready (scheduled) to run
DEAD	this process is unable to run, but still holds resources e.g. zombies and **Debugger32**
SEND	this process has sent a message to another process which hasn't *Receive()*d it yet
REPLY	this process has sent a message to another process which has *Receive()*d it, but not *Reply()*d to it
RECV	this process has called *Receive()* and is waiting for a message
HELD	this process was signalled with SIGHOLD and will not become READY until it is signalled with SIGCONT
WAIT	this process is *wait()*ing for a death status of a child or waiting for a timer to expire (e.g. *sleep()*ing)

continued...

State	Description
SIGBL	this process has sent a system signal message (because it was signalled while REPLY blocked on a process which has the **_PPF_SIGCATCH** process flag set) which hasn't been *Receive()*d yet.
SEM	this process is waiting to procure a semaphore

Current Directory

d The **d** (for "directory") format character causes a CURRENT DIRECTORY heading and each process's **$PWD** environment variable to be displayed. A "Not available" message is displayed if you do not have appropriate permissions.

Environment Variables

e The **e** format character causes an ENVIRONMENT heading and each process's environment to be displayed. If you do not have appropriate permissions, a "Not available" message will be displayed under the ENVIRONMENT heading. If the environment list exceeds 512 characters, sin will truncate the list and display a " ... " at the end to signify that truncation occurred.

Process Flags

F The **F** format character causes a FLAGS heading and each process's process flags to be displayed. The following flags are currently defined: (see *qnx_pflags()* for more information)

```
fSZHTLE mMVXF3 +    DSmcsfr FiI
```

Process flags which are not set will appear as dash (**-**) characters e.g.:

```
-SZ----------3-------S------FiI
```

Char	Meaning
f	Forked
S	Session leader
Z	No-Zombie
H	Held (or to be held)
T	Terminating (or zombie)
L	Loading
E	Execing (transforming)
m	Remote proxy
M	Proxy
V	Virtual circuit
X	Execute-in-place
F	Flat model
3	32-bit
+	Was 32 bit
D	Rdos (Older versions of QNX only)
S	Server
m	Sig mask all
c	Sigcatch
s	SIGCHLD ignored
f	Priority float

continued...

© 2005, QNX Software Systems

Char	Meaning
`r`	Priority receive
`F`	Fixed segments
`i`	Informed
`I`	Immortal

Family (Dad/Son/Brother Relationships)

f The `f` format character causes a DAD, SON and BRO heading and each process's parent PID, a child PID, and another PID which has the same parent PID, respectively, to be displayed.

Process Group ID

g The `g` format character causes a PGRP heading and each process's group ID (GID) number to be displayed.

Process ID

i The `i` format character causes a PID heading and each process's ID number to be displayed.

Login Name

L The `L` format character causes a USER NAME heading and the user that started the process for each process to be displayed.

Magic

M The `M` format character causes a MAGIC heading and the magic data selector and offset of each process to be displayed.

Size

m The **m** format character causes a CODE and DATA heading and the size of each process's code and data memory spaces to be displayed. The method **sin** uses to calculate these values was changed, and may change again. Currently, **sin** tries to determine the amount of memory unique to/or really used by each process. Overlaid and mapped (32-bit only) memory just gives access to "other" memory regions, and are not accounted for. Shared DATA is accounted for in the process which owns the segment, not a process which copies the selector.

A CODE size of 0 indicates that this process is 16 bit and is sharing its CODE with another instance of the same process (e.g. **ksh**). The first instance of this process will have the full CODE size allocated to it. With 32-bit processes, code sharing causes CODE to become

(*total code size***)** ÷ **(***number of instances***).**

Session node ID

N The node ID the session was originally started in.

Process Name

n The **n** format character causes a PROGRAM heading and the name of each process to be displayed. The **sin** utility formats the PROGRAM field to 23 characters. If the name is longer, the node prefix and as much of the basename as possible will be displayed. A ***** indicates that (middle) truncation occurred. For example,

```
//node/dir1/dir2/dir3/name
```

would be truncated to:

```
//node/*/dir3/name
```

NOTE: a name of **(zombie)** indicates this process has terminated and its parent hasn't *wait()*ed for its exit status yet.

Memory Pass

P The **P** format character causes the MPID MSEL and MFLG headings and the segment arming information of each process to be displayed. These headings directly map into the parameters to the *qnx_segment_arm()* library routine. e.g. **qnx_segment_arm(MPID, MSEL, MFLG)**

Priority

p (priority) The **p** format character causes a PRI heading and the priority and scheduling algorithm of each process to be displayed. The scheduling algorithm is represented by one of the following 3 characters:

Char	Meaning
r	round robin
f	FIFO
o	other (adaptive)

Pathname Prefix Root

R The **R** format character causes a ROOT heading and each process's root prefix to be displayed. Usually, this is simply the node ID. Unless you are logged in as **root**, you can obtain the pathname prefix roots only for processes being run by your own user ID.

Registers

r The **r** format character causes each process's Intel registers to be displayed. For 32-bit processes, a second line of output is needed to display the most-significant part of each register.

Signals

S The **S** format character causes the following headings to be displayed:

Heading	Description
SIG IGN	signals which are set to be handled by SIG_IGN
SIG MASK	signals which are currently masked (blocked)
SIG PEND	signals which are pending
SEG:OFF	*segment*:*offset* of the signal table

There are 32 signals. Each signal is represented as a bit within the 4 byte field, with signal 1 being the LSB. For example, signals 1 and 5 would be represented by: (0x)**00000011**

The signal table consists of 2 **unsigned**s followed by 32 **struct _sigaction** structures (1 for each signal).

Session ID

s The **s** format character causes a SID heading and each process's session ID to be displayed.

File Time and Execution Start Time

T The **T** format character causes a FILE TIME and a START TIME heading to be displayed. FILE TIME is the time stamp on the process's executable file (**ls -l** *file*) and START TIME is the date/time when this process was started.

CPU Time Used

t The **t** format character causes the following headings to be displayed:

Heading	Description
START TIME	date/time when this process was started
UTIME	amount of CPU seconds this process has consumed
STIME	amount of CPU seconds the system has consumed on behalf of this process (e.g. **Proc**)
CUTIME	amount of CPU seconds children of this process have consumed.
CSTIME	amount of CPU seconds the system has consumed on behalf of the children of this process.

Children times are added to CUTIME and CSTIME only after they terminate.

The **UTIME**, **STIME** and **CUTIME** fields will be zero if the process manager (**Proc**) has been started with the **-a** option.

Real User ID and Group ID

U The **U** format character causes a RUID and a RGID heading to be displayed. RUID is the real user ID and RGID is the real group ID of each process.

Effective User ID and Group ID

u The **u** format character causes an EUID and an EGID heading to be displayed. EUID is the effective user ID and EGID is the effective group ID of each process.

Other process information queries

File descriptors — `sin fds`

The **`fds`** query is useful for displaying which files and devices a process has open. These are displayed under the process name and PID, 2 per line. The format for each fd is:

fd number **`-`** *file path*

If the fd number has an adjacent "**`C`**", the _FD_CLOSE_ON_EXEC flag is set for that fd. The **`sin`** utility determines the file path by using the *dev_fdinfo()* library routine first. If this call succeeds, the returned device path is used. If the path is not set, the returned device type is used and is displayed inside brackets (e.g. **`[con]`**). If the call to *dev_fdinfo()* fails, *fsys_fdinfo()* is called. If this call succeeds, the returned file path is used. If the *fsys_fdinfo()* call fails, the PID of the fd's manager is displayed inside parentheses (as a last resort).

There are some words which may appear in the place of a file path to indicate an unusual status of the file which prevents **`sin`** from obtaining a filename. These are:

A file path of:	Indicates that:
`unknown`	**`sin`** could not establish a VC to a remote manager.
`unlinked`	the file has been removed (unlinked) but this process still holds it open.
`pipe`	indicates the file is an unnamed FIFO (pipe)
`[@1234567]` etc.	indicates the file is really a socket (**`Socket`** manager)

Files/pathnames in use — `sin files`

The **`files`** query displays the following headings:

© 2005, QNX Software Systems

Heading	Description
PROGRAM	process name
PID	process ID number
FD	file descriptor number
MODE	open mode flags
POSITION	current position/size
FILE	fullpath

The following mode flags are displayed if set:

Flag	Meaning
E	Exclusive
F	File Sync
D	Data Sync
A	Append
W	Write
R	Read

If the flag is not set, a dash (**-**) will appear in its column instead.

Memory regions — `sin memory`

The **`memory`** query displays each process's selectors 2 per line. The format is selector, address, limit and flags. The following flags may be displayed:

```
LBS3+++GA+PMDCO
```

which mean, in order:

Flag	Meaning
L	Process is loading into this segment
B	Borrowed, segment was taken from the free list
S	Shared, more than one process owns this segment
3	Segment is code/stack segment in 32-bit mode
+	Segment is a link to an existing segment
+	Allow DMA requests above 16Meg
+	Disable caching if possible
G	Global segment, is in GDT
A	Align segment on a GET/PUT operation
+	Global segment and is owned by this process
P	Huge segment, next segment continues this one
M	Process can modify access rights
D	DMA safe, will not cross 64k segment boundary
C	Code readable/executable
O	Data read only

Note: If both “Data read only” (**O**) and “Code readable/executable” (**C**) are shown, then this is really “Code execute only” (**CO**). Also, if neither “Data read only” nor “Code readable/executable” are shown, then this is really “Data readable/writable”.

Other OS resource information queries

The second class of queries supported by **sin** reports on most of the major operating system objects, such as interrupt handlers and proxies.

Free memory — `sin freemem`

The **`freemem`** query simply prints out each contiguous block of free memory. Note that for most memory requirements, physically contiguous blocks of memory are not required because each process operates in a virtual address space which is mapped into physical addresses through a page table.

Global names — `sin gnames`

The **`gnames`** query displays all the registered global names and the node on which they were registered from. This information is returned from a **`nameloc`** process running on the network.

Registered names — `sin names`

The **`names`** query displays all names registered on the node, as well as the PID and program name of the process which registered the name. These include not just local names, but also the global names registered on this node.

Proxies — `sin proxies`

The **`proxies`** query displays the following headings:

Heading	Meaning
PROXY	the proxy ID (PID of the proxy)
PROGRAM	the process name which owns the proxy
PRI	the priority of the proxy
STATE	the state of the proxy (READY or SEND)
COUNT	the pending amount of triggers

Note that a proxy becomes SEND blocked when the COUNT becomes greater than 0, and the owning process hasn't *Receive()*d the proxy message yet.

Realtime timers — `sin rtimers`

The **`rtimers`** query displays information about the realtime timers. The following headings are displayed:

Heading	Meaning
ID	the timer ID
PID	the PID of the process which owns the timer
PROGRAM	the name of the process which owns the timer
ACTION	the "purpose" of the timer (what it does when expired). This will be one of **`sleep`** (meaning that the process is blocked until the timer expires), **`proxy`** *PID* (meaning that the proxy *PID* will be triggered on timer expiry), or **`signal`** *SIG* (meaning that the *SIG* signal will be raised on the process on timer expiry.)
TRIGGER	the seconds before the timer expires
REPEAT	the seconds between each TRIGGER

Sessions — `sin sessions`

The **`sessions`** query displays information about the current sessions. The following headings are displayed:

Heading	Meaning
SID	the session ID
USERID	the user which owns the session
DEVICE	the controlling terminal of the session
PID	the PID of the session leader

continued...

Heading	Meaning
MEMBERS	the number of processes in the session

Virtual circuits — `sin vcs`

The **`vcs`** query displays information about the virtual circuits (VC) of a node. Each VC has 2 ends, LOCAL and REMOTE.

Under the LOCAL heading, the following headings are displayed:

VID
: the VC ID

PID
: the process which owns the VID

LK
: the link count (when a PID has multiple open fds to the same remote manager, the VC is shared — instead of a VC for each fd)

STATE
: state/sub-state of this VC end. Possible values are:

State	Meaning
READY/1	idle or txing reply
REPLY/2	initializing VC
REPLY/3	initializing existing VC for sharing
REPLY/4	growing on *Send()*, blocked on local **`Proc`**
REPLY/5	growing on *Send()*, blocked on **`Net`**
REPLY/6	sending, blocked on network
REPLY/7	growing on *Reply()*, blocked on local **`Proc`**
REPLY/8	growing on *Reply()*, blocked on **`Net`**

continued...

State	Meaning
REPLY/9	growing on *Writemsg()*, blocked on local **Proc**

PROGRAM the name of the process which owns the VID

Under the REMOTE heading, the following headings are displayed:

NID	the node ID of the remote end of the VC
VID	the VC ID on the remote node (NID side)
PID	the process which owns the VID on NID
PROGRAM	the name of the process which owns the VID on NID

General system information

The third class gives reports on important but more general information, for example, the versions of the major System Manager processes running.

Global descriptor table — `sin gdt`

The **gdt** query walks down the Global Descriptor Table and displays each entry. This information is very Intel-specific and is beyond the scope of this document. You must be **root** to use **sin gdt**.

Interrupt descriptor table — `sin idt`

The **idt** query walks down the Interrupt Descriptor Table and displays each entry. Similar to the **gdt** query, this information is very Intel-specific and is beyond the scope of this document. You must be **root** to use **sin idt**.

General system information — `sin info`

The **`info`** query displays information about a node's hardware and configuration. The following headings are displayed:

NODE	logical node number
CPU	type of CPU/FPU
MACHINE	type of machine (AT, PS/2, PCI,...)
SPEED	a benchmark result performed at boot up
MEMORY	available **/** total memory
TICKSIZE	current ticksize
DISPLAY	type of monitor (None, Monochrome, CGA, EGA Color, EGA Mono, PGS, VGA Mono, VGA Color, PS/2 30 Mono, PS/2 30 Color)
FLAGS	the following system flags:

```
D3P+-------AEE8P
```

where the flags mean (in order):

Flag	Meaning
D	memory reserved for Rundos (older QNX only)
3	Supports 32-bit processes
P	PCI Bios
+	32-bit Kernel
-	(unused)
A	APM installed

continued...

Flag	Meaning
E	Emu32 installed
E	Emu16 installed
8	NDP installed
P	Protected

HEAPP	**Proc**'s free heap space
HEAPF	**Proc**'s free file descriptor heap space
HEAPL	**Proc**'s free LDT/GDT heap space
HEAPN	**Proc**'s free registered name space
HANDS	maximum number of interrupt handlers
NAMES	maximum number of names (local and global)
SESSIONS	maximum number of sessions
PROCS	maximum number of processes
TIMERS	maximum number of timers
NODES	number of QNX licenses
VIRTUAL	Free virtual memory/Total virtual memory
BOOT	date and origin of boot (a "*****" indicates alternate boot)
LOCATORS	A list of nodes running **nameloc** from which global names, network alive lists and network licensing information will be fetched.

Interrupts used — `sin irqs`

The **`irqs`** query displays which interrupts are in use and by which processes. The following headings are displayed:

IRQ	the hardware interrupt vector (-1 means special 50 millisecond timer)
PID	the process which attached the interrupt handler to the IRQ
PROGRAM	the name of the process which attached the interrupt handler to the IRQ
CS:IP	the selector:offset of the interrupt handler
DS	the data selector used by the interrupt handler

Information about nodes on the network — `sin net`

The **`net`** query displays information about each node on the network (that can be reached). The following headings are displayed:

NID	logical node ID
MACHINE	type of machine (AT, PS/2, PCI)
CPU	type of CPU
FPU	type of FPU
SPEED	a benchmark result performed at boot up
MEMORY	total memory
HARD	non-removable hard disk space present under **`/dev`** with a block-special filename consisting of an alphabetical name followed by a number (e.g. **`/dev/hd0`** would be counted but **`/dev/hd0t77`** would not).

FLOPPY	the size of **/dev/fd0**
OTHER	indicates the presence of some other facility, such as

TCP	the node has **Socket** running
cd	the node has a CD-ROM drive
rhd	the node has a removable hard disk drive
tp	the node has a tape drive

DISPLAY	type of monitor (None, Monochrome, CGA, EGA Color, EGA Mono, PGS, VGA Mono, VGA Color, PS/2 30 Mono, PS/2 30 Color)
FLAGS	system flags (see the **info** query)

SPEED, MEMORY, and HARD values are totalled for the network and are displayed on the last line of output.

Process family tree — `sin tree`

The **tree** query displays a "family tree" of the processes. Children are shown to the right of their parents. Siblings are shown vertically (all connected to each other and their parent).

Server process versions — `sin versions`

The **versions** query displays the version and date of each server process (process flag **_PPF_SERVER** set). This information is returned from a system version message sent to each server process.

Examples:

Show all processes running on the local node in default format:

```
sin
```

Show process times for all processes in session 0 on the local node:

```
sin -s 0 time
```

© 2005, QNX Software Systems

Show only those processes running in your own session:

```
sin -m
```

Show general system information for node 4:

```
sin -n 4 info
```

Show registers for process 4 on node 5:

```
sin -n 5 -p 4 registers
```

For each process running on the local node show process ID, effective user ID and group ID, real user ID and group ID, and process name:

```
sin format iuUn
```

Show all processes running on the local node in the default format, sorted by priority:

```
sin -S p
```

See also:

ps, **sac**

Syntax:

sinit [-i *initfile***] [-r** *root***] [-s** *shell***]**
[*var***=***value***]...**

Options:

-i *initfile*	Open this system initialization file (for defaults, see the Description below).
-r *root*	Define an alias prefix for the filesystem root (**/**).
-s *shell*	Invoke this shell (default is **/bin/sh**).
-w	Wait for a name locator to communicate with our node. Only useful for disk boot. See also **nameloc -k**.
*var***=***value*	Define an environment variable.

Description:

The **sinit** utility sets up the initial operating system environment immediately after boot. It is usually bound into the operating system image by means of the **buildqnx** utility.

To run successfully, **sinit** needs to access a filesystem that contains a shell that can process the system initialization file. This filesystem can be local (a hard disk on that machine) or remote (a hard disk on another machine). If the filesystem is local, the image must contain the Filesystem Manager (**Fsys**) and a disk driver to access the local disk. If the filesystem is remote, the image doesn't need the Filesystem Manager, but will require the Network Manager (**Net**) and an appropriate network driver to access the remote filesystem over the network.

The **sinit** utility attempts to find and open the system initialization file for your node. This is a text file that contains a list of system initialization commands for which **sinit** invokes a shell to execute.

There are two boot types: a *normal boot* and an *alternate boot*. All boots over the network are considered normal. When you boot from

disk, you are prompted with the option of doing an alternate boot. The type of boot you choose determines which system initialization file will be used.

If you choose:	**sinit** tries to find and open:
a normal boot	**/etc/config/sysinit.***node* If this fails, it tries: **/etc/config/sysinit**
an alternate boot	**/etc/config/altsysinit** If this fails, it tries: **/etc/config/sysinit**

It's usually a good idea to keep the **sysinit** file generic and create custom **sysinit.***node* files for each node as required. The **altsysinit** file is usually a backup of the **sysinit.***node* file for the node that boots from its own hard disk.

Whenever you wish to get your files from another node (remote file server), use the **-r** *root* option. If you boot from a hard disk and get your files from a local hard disk, you don't need to specify this option.

You can override the default search for **sysinit**.*node* or **altsysinit** by specifying an explicit filename with the **-i** *initfile* option.

The default shell invoked by **sinit** is **/bin/sh**. To override this default, use the **-s** *shell* option. You should specify a complete pathname for the shell, starting with a slash (**/**).

You can cause **sinit** to create an environment variable using the *var***=***value* operand. These environment variables will be inherited by the system initialization shell. The time zone variable (**TZ**) is often set here.

The **sinit** utility always terminates; it doesn't stay resident in the system. User **login**s etc. are started by the terminal initialization utility, **tinit**, which should be started in the **sysinit** file for the machine in question.

Examples:

Initialize the system, using the filesystem on node 1 instead of a local disk (which may be absent):

```
sinit -r //1/
```

Initialize the system using the shell **/bin/myshell**:

```
sinit -s /bin/myshell
```

See also:

buildqnx, **tinit**

Installation & Configuration

size

© 2005, QNX Software Systems

Print section sizes of object and load modules (UNIX)

Syntax:

size [-v] *file*...

size [-h *heap***] [-s** *stack***]** *module*...

Options:

-h *heap*	Modify initial heap size to *heap*.
-s *stack*	Modify stack size to *stack*.
-v	Verbose display; includes stack and heap information.

Description:

The **size** utility displays size information for each section in the specified object or load modules.

The **-s** *stack* option will not work for 32-bit flat model objects or load modules.

The output consists of five columns. With the exception of the hex column, all information is printed in decimal.

This column:	Indicates:
text	the total code size in bytes
data	the total initialized data size
bss	the total uninitialized data size (this includes the stack, the heap, and any other uninitialized data items)
decimal	the sum of text, data, and bss
hex	the sum of text, data, and bss (in hex)

Examples:

Entering this command:

```
size /bin/size /bin/wcc
```

would display a table similar to the following:

```
text    data    bss     decimal   hex
6816    620     6276    13712     3590    /bin/size
353276  31832   20796   405904    63190   /bin/wcc
```

Exit status:

0 Successful completion.

>0 An error occurred.

slay

© 2005, QNX Software Systems

Kill or modify a process by name (QNX)

Syntax:

slay [*options***]...** *process_name***...**

Options:

- *signal_number*

A signal number specifying which signal to raise on the processes matching *process_name*s.

-f Force the action to be taken on all processes sharing the same *process_name*. Normally, **slay** prompts for confirmation when more than one process bears the specified name.

-h Set a SIGSTOP signal on a process, effectively holding its execution.

-n *nid* Search for the specified processes on the remote node *nid*.

-P *prio*[**f|r|o**] Set the processes matching *process_name* to priority *prio*. Non-**root** users are limited to a maximum priority of 19. The priority may be followed by **f**, **r** or **o** to change the scheduling policy to FIFO, RR or OTHER, respectively.

-q Query before dealing with the process, even if only one process is found with a matching name (overrides option **-f**). This option is useful for viewing the other process information that **slay** presents.

-Q Be quiet. This option is useful when **slay** is invoked from a C program.

-s *sig* Define a signal to send. This option causes the signal *sig* to be raised for the processes matching the *process_name*s.

-S	Suppress killing processes that have child processes. A typical use of this command would be within a shell command that shuts down shells on other devices. Setting this option would prevent **slay** from killing those shells that had other processes (such as editors) running. If **-q** is also specified, **slay** will prompt for a forced kill even if the named process has child processes.
-t *ttyname*	Match only those processes whose name is *process_name*, with *ttyname* as the controlling terminal. If *ttyname* doesn't begin with a slash (**/**), **slay** assumes that it starts with the **/dev/** prefix.
-u	Set a SIGCONT signal on a process. If execution of the process was being held by a SIGSTOP signal, execution will begin where it left off. If the process had not previously had a SIGSTOP set upon it, the SIGCONT signal has no effect.
-v	Be verbose; display messages about processes being signaled.
process_name	The name of a process to operate on.

Description:

You use the **slay** utility to kill a process by name rather than by process ID. This saves you from having to first run the **ps** utility to obtain a pid before issuing a **kill** command. Process names are specified without the path. For example, let's say you have a process called **/bin/sleep** that you wish to kill. Entering **sleep** as the process name would be sufficient to allow **slay** to find and kill it.

There are many forms of this command. The simplest and most often used form is:

```
slay process_name
```

© 2005, QNX Software Systems

This command locates the process bearing the specified name. If only one is found, a SIGTERM signal is set on it. If more than one process bears the specified name, you are prompted for a yes/no response for each process. When each process is listed in this form, the process name, *pid*, and tty group/member numbers are also displayed to help you make a selection.

To set a signal on a process you must either own the process or be logged in as **root**.

Examples:

Kill **spooler** process on node 2:

```
slay -n 2 spooler
```

As **root**, change priority of the **test** process to 20:

```
slay -P 20 test
```

Exit status:

0	No processes matched the supplied criteria, an error occurred, or the number of processes matched and acted upon was an even multiple of 256.
1-128	The number of processes matched and acted upon modulo 256 (e.g. a status of 1 could mean 1 process, 257 processes, 513 processes etc.)
129-160	If the exit status was gleaned through direct spawning, this is the number of processes matched and acted upon modulo 256. If **slay** was run through the shell, this will be either the number of processes matched and acted upon, or will be the cause of **slay** dying due to a signal (subtract 128 from the exit status to determine the signal number).

161-255 The number of processes matched and acted upon, modulo 256.

Caveats:

The exit status of **slay** is non-standard for historical reasons. It is strongly recommended that **slay** not be used in any situation where the exit status will be relied upon because the status is ambiguous in some circumstances.

See also:

kill, **ps**, **sin**

sleep

© 2005, QNX Software Systems

Suspend execution for an interval (POSIX)

Syntax:

sleep *time*

Options:

time A non-negative decimal integer specifying the number of seconds to suspend execution. Note that very large values for *time* that would sleep past the year 2037 should be avoided. Note also that values greater than 65536 are not necessarily supported on all POSIX systems.

Description:

The **sleep** utility suspends execution for at least the integral number of seconds specified by the *time* operand.

Examples:

Sleep for 15 seconds:

```
sleep 15
```

Signals:

The **sleep** utility takes the default action for all signals, except that **sleep** terminates normally with a zero exit status on receipt of a SIGALRM signal.

Exit status:

0 The execution was successfully suspended for at least *time* seconds, or a SIGALRM signal was received.

>0 An error occurred.

See also:

sh

Slib

© 2005, QNX Software Systems

System shared library (QNX)

Syntax:

Slib32

Slib16

Options:

None.

Description:

For QNX to operate, a system shared library must be present in the operating system image that is booted. There are two of these libraries:

- **Slib32** — allows 32-bit programs to run. This library is mandatory.
- **Slib16** — allows 16-bit programs to run. This library is mandatory to run any 16-bit programs.

If you want to run both 16- and 32-bit programs on your system, you *must* run both shared libraries. Always place the 32-bit shared library *before* the 16-bit one in the boot image.

Examples:

The following build file would be for a node using an Ethernet 1000 card, where only 32-bit programs could run:

```
sys/Proc32
$ Proc32 -l $(lnode)

sys/Slib32
$ Slib32

/bin/Net
$ Net -m $(netmap)

/bin/Net.ether1000
```

```
$ Net.ether1000

/bin/sinit
$ sinit -r //$(bnode)  TERM=qnx  TZ=$(TZ)
```

Same as above but for a node where both 16- and 32-bit programs could run:

```
sys/Proc32
$ Proc32 -l $(lnode)

sys/Slib32
$ Slib32

sys/Slib16
$ Slib16

/bin/Net
$ Net -m $(netmap)

/bin/Net.ether1000
$ Net.ether1000

/bin/sinit
$ sinit -r //$(bnode)  TERM=qnx  TZ=$(TZ)
```

See also:

buildqnx, **Proc**

Installation & Configuration

SMBfsys

© 2005, QNX Software Systems

SMB Filesystem Client (QNX)

Syntax:

SMBfsys [-d *nminutes***] [-i** *size***] [-N] [-n** *name***]**
[-o] [-p *priority***] [-P [~]***semantic***]**
[-r *nseconds***] [-R** *bufsize***] [-S** *bufsize***]**
[-u *user***] [-x s|h|a] &**

Options:

-d *nminutes* — If no requests are made for a particular server after *nminutes* minutes, **SMBfsys** detaches itself from that server to save resources. It automatically reattaches itself the first time a new request for the server comes in. Note that many SMB servers automatically detach themselves from a client if they don't receive a request from them within a certain time period as well. **SMBfsys** automatically reattaches in either case. The default for this option is 60 minutes.

-i *size* — Set the size of the inode cache in kilobytes. Default is 1K; if a value of **0** is specified, inode values are simply a hash function of the filename thus don't require a cache. The downside of this is that multiple files end up with the same inode number. (See description.)

-N — Turn off TCP_NODELAY socket option (default is on). Performance to NT servers is helped when TCP_NODELAY is set on.

-n *name* — Set the NetBIOS name of the client. Since **SMBfsys** uses NetBIOS over TCP/IP to send requests, the client actually has two names — the TCP/IP name, as given by DNS or the **/etc/hosts** file, and the name used by the NetBIOS session that **SMBfsys** establishes with the server. By default, **SMBfsys** uses the TCP/IP name, up to the first dot as the

NetBIOS name. If this option is used, **SMBfsys** uses *name* as the NetBIOS name instead.

-o When a request comes in from a QNX user ID that **SMBfsys** hasn't seen before, it attempts to establish a new user logon with the SMB server using the information provided by the **user_smb** program. If the SMB server refuses the connection because it says that it has too many users from this client (NT Workstation appears to allow only one user from a particular client machine to be logged on at one time), **SMBfsys** normally returns an error condition to the requesting process. If **-o** is specified, **SMBfsys** instead uses the logon connection of the original process to make a request of the SMB server (this is the user ID of the process that originally mounted the server). That means that if QNX user ID 5 first makes a request of an SMB server and gets logged in, then later QNX user ID 7 tries to make a request and its logon is refused, **SMBfsys** uses user ID 5's connection to satisfy the request. This may give user ID 7 different permissions than expected to the SMB server's file system.

-p *priority* Set the priority that **SMBfsys** is to run at. The default is 15.

-P [**~**]*semantic*

Enable (or if prefixed with **~**, disable) certain aspects of POSIX filesystem semantics. Allowed semantic names:

delete_permission

POSIX says permission to delete a file is given by write permission on the parent directory. Normal DOS semantics gives delete permission if the file itself is writable.

delete_open POSIX says you can delete a file that a process has open. The contents remain valid until the last close. DOS semantics says it's an error to do this (though some servers actually make the deletion and just lose track of the space). SMBfsys tries to fake this by just setting a bit internally to delete the file after everybody's closed it. This can fail if a client on a different machine has it open. The filename also remains visible in the directory so other processes can open it after POSIX says it should be inaccessible.

rename_to_existing POSIX says you should unlink an existing file name before renaming a file to it. Under DOS semantics it's an error to attempt to rename a file to a name that already exists.

The defaults in all cases are to use the standard DOS semantics.

-r *nseconds* If **SMBfsys** makes a request of an SMB server and hasn't received a response after *nseconds* seconds, it deems the server to have gone down and reports a failure indication to the requesting process. The default is 30 seconds.

-R *bufsize* Specify the size of the socket receive buffer in kbytes.

-S *bufsize* Specify the size of the socket send buffer in kbytes.

-u *user* When a user ID comes in that **SMBfsys** has no **user_smb** mapping information for, it attempts to

read the **/etc/passwd** file to obtain a user name to give to the SMB server. If **SMBfsys** can't find the user ID in **/etc/passwd**, it uses *user* as the username to be given to the SMB server. The default is **guest**.

-x s|h|a Use the DOS system (**s**) or hidden (**h**) attributes to indicate POSIX execute permission on a file, or have all (**a**) files executable (system and hidden aren't used). The default is to use the system attribute.

Note that the DOS/Windows/NT **DIR** command doesn't display files with the system or hidden attributes by default. That means that making a file executable on the QNX side causes it to apparently vanish on the server side. It's still there and you can open it if you know the name, it just doesn't show up in directory listings. To see it under DOS/Windows/NT you either have to use **DIR /A** or use some other command that shows files with those attributes.

Description:

The SMB (Server Message Block) file sharing protocol is used by a number of different servers such as Windows NT, Windows 95, Windows for Workgroups, LanMan, and Samba. **SMBfsys** allows a QNX system to access remote drives residing on such systems as an SMB client.

SMBfsys implements this protocol using NetBIOS on TCP/IP only, not NetBEUI. Accordingly, you need TCP/IP installed on both the QNX and remote server side. Once **SMBfsys** is running and a remote server has been mounted, the server's file system appears as a sub-tree in the local directory structure.

The **/etc/services** file must have the following services in it :

```
netbios-ns    137/udp    # NetBIOS name server
netbios-dgm   138/udp    # NetBIOS datagram service
```

```
netbios-ssn   139/udp    # NetBIOS session service
netbios-ssn   139/tcp
```

Once **SMBfsys** is running, you must perform two more steps to mount a remote filesystem. First, you need to tell **SMBfsys** your remote user name and password. This is done with the **user_smb** command. Once this is done, you can mount the remote filesystem with the SMB mount command (**mount_smb**).

POSIX has the concept of a file inode, or internal node value. These are unique numbers for each data file (two hard-linked files share the same inode value). For the most part, file inode values are unimportant, but certain programs do look at them (e.g. **cp** uses them to determine if the source and destination are the same file). There's no corresponding value available from an SMB serve,r so **SMBfsys** has to fake a value. An inode counter is incremented every time someone makes a request that returns an inode value. However, this means that the inode value for a file would be constantly changing.

To partially alleviate this, **SMBfsys** maintains a cache of recently requested files and their returned inode value. If a request for an inode value comes in, **SMBfsys** checks its cache first and if the filename is present, the previously returned value is used. Since file names are of variable length, the number of files in the cache depends on how long the names are. Once a filename has been flushed from the cache, the next time the inode value is requested a different number is returned. If the inode cache size is set to zero (**-i 0**), **SMBfsys** uses a different algorithm. In this case, it takes the file name and performs a hash function on it. The result of the hash function is used as the inode value. This means that the inode value for a given file always remains constant, but two different files might return the same inode value if they hash to the same number.

Examples:

Run **SMBfsys** using default parameters:

```
SMBfsys &
```

Disconnect from a server if no requests have been made for 15 minutes and use **FOOBAR** as the NetBIOS name of the client.

```
SMBfsys -d 15 -n FOOBAR &
```

Files:

/etc/services

The service-name database.

Caveats:

SMBfsys requires a socket manager.

See also:

Dosfsys, **Fsys**, **mount**, **mount_smb**, **umount**, **user_smb**

Socket, **Socklet** in the *TCP/IP User's Guide*

sort

© 2005, QNX Software Systems

Sort, merge, or sequence-check text files (POSIX)

Syntax:

sort [-m] [-o *name***] [-bdfinru] [-t** *char***] [-k** *keydef***]**
[+*oldkey***] [***file***...]**

sort [-c] [-bdfinru] [-t *char***] [-k** *keydef***] [+***oldkey***]**
[*file***]**

Options:

-c Check that the single input file is already sorted. Produce no output.

-m Merge only. It's assumed that the input files are already sorted.

-o *name* The *name* argument is the name of an output file to be used instead of the default, which is standard output. This file can be the same as one of the input files.

-u Unique; suppress all but one key in each set of lines having equal keys.

The following options override the default ordering rules. When ordering options appear independent of key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key (see option **-k**), the specified ordering options override all global ordering options for that key.

-b Ignore leading **blank** characters in field comparisons.

-d Sort in "dictionary order." Only **blank** characters and alphanumeric characters are significant in comparisons.

-f Fold uppercase letters into lowercase.

-i Ignore all non-printable characters.

-k *keydef* Define *keydef* to be a sort key.

-n	Interpret the field as numeric, including the sign and optional “thousands” separator. Sort in numeric order. This implies the **-b** option.
-r	Reverse the sort order.
-t *char*	Use *char* as the field separator character.
file	The pathname of a file to be sorted, merged, or checked. If you don’t specify any files to be sorted, or if a *file* operand is the dash character (**-**), the standard input is used.
+*oldkey*	This is a historical key-field mechanism. For more information, see the Description.

Description:

The **sort** utility orders lines from a set of files (or standard input if no files are provided) and merges the output into the specified output (or standard output if no output file is specified). The **sort** utility provides a number of options for controlling the sorting mechanism. The default behavior of **sort** is to treat all lines as a sequence of characters to be sorted in ascending order.

The **sort** utility regards files and file contents the following way:

- a file is a sequence of lines
- a line is a sequence of fields followed by a **newline**
- a field is a sequence of characters followed by a field separator
- a field separator, by default, is a **blank**, although this may be altered by the **-t** option

The **-t** option has subtle effects. If the option is *not* used, only the first **blank** in a string of **blank** characters is interpreted as a field separator; the remaining are considered part of the next field. If the **-t** option is used, every occurrence of the field separator character is

considered a terminator for a field. Thus, four adjacent field separators represent three empty fields.

A sort key field may be defined using the following syntax:

-k *field_start* **[***type_string***] [,***field_end***] [***type_string***]**

where *field_start* and *field_end* specify the beginning and end of the key field, and *type_string* is used to specify attributes specific to the key.

The *field_start* and *field_end* are each specified by a pair of digits of the form *m***.***n*, where the *m* refers to the field starting after the *m*th field separator in a line. For *field_start*, the **.***n* refers to the *n*th character of the specified field, and is taken as zero if not specified. For *field_end*, the **.***n* refers to the *n*th character after the last character of the specified field, and is taken as zero if not specified.

The *type_string* may be formed from the characters **bdfinr** which apply their defined attributes to the determination of the key.

There is another mechanism (**+***oldkey*) for defining sort keys, which remains for historical reasons. The syntax of this mechanism is:

+*field_start* [*type_string*] **-***field_end* [*type_string*]

The semantics are the same as for the **-k** option.

The following command would sort the password database by *gid*, then *uid*, then *username*:

```
sort -t: -k 3.0n -k 2.0n -k 0.0d /etc/passwd
```

Notice that the *type_string* attached to *field_string* is used to determine the type of comparison with this key.

The second *type_string* is solely to adjust the extent of the *sort_key*, thus only **i** and **b** make any sense. Any other characters are ignored.

Examples:

The standard password file has the form:

username:*password*:*uid*:*gid*:*misc*:*home*:*shell*

To sort the password database by *username*:

```
sort -t: /etc/passwd
```

Exit status:

0	All input files were output successfully, or option **-c** was specified and the input file was correctly sorted.
1	The **-c** option was used, and the input file was not sorted.
>1	An error occurred.

See also:

diff, **grep**, **sed**

spatch

© 2005, QNX Software Systems

Fullscreen patch utility (QNX)

Syntax:

spatch [-bp] *file* **[***offset***]**

spatch -m *pid selector offset*

Options:

-b Browse only: don't allow the **Save** command. The file or disk will be accessed in read-only mode.

-m *pid selector offset*

Examine RAM for the specified process, where *pid* is a process ID and *selector* can be either an LDT or a GDT selector.

-p Pause before starting, for example to allow a floppy disk to be inserted in a disk drive.

file The file or disk to be examined.

offset The address — in RAM, in the file, or on the disk — where the **spatch** is to begin (specified in hex).

Description:

The **spatch** utility provides fullscreen editing of files or disk blocks. The screen displays a 16-by-16 (256) byte image of the data being examined, similar to that shown here:

```
 Edit Next Prev Lastblk Home Goto Find Continue Save Addr Quit
000000000: 2E 28 6E 65 77 29 20 53  50 41 54 43 48 20 22 46 .(new) SPATCH "F
000000010: 75 6C 6C 20 73 63 72 65  65 6E 20 70 61 74 63 68 ull screen patch
000000020: 20 75 74 69 6C 69 74 79  22 1E 2E 28 73 79 6E 74  utility"..(synt
000000030: 61 78 29 1E 09 11 73 70  61 74 63 68 10 20 20 11 ax)...spatch.  .

000000040: 66 69 6C 65 10 20 20 AE  66 69 6C 65 6E 61 6D 65 file.  .filename
000000050: AF 1E 09 11 73 70 61 74  63 68 10 20 20 11 64 69 ....spatch.  .di
000000060: 73 6B 10 20 20 AE 64 72  69 76 65 AF 20 20 AE 62 sk.  .drive.  .b
000000070: 6C 6F 63 6B AF 1E 09 11  73 70 61 74 63 68 10 20 lock....spatch.

000000080: 20 11 6D 65 6D 10 20 20  AE 73 65 67 6D 65 6E 74  .mem.  .segment
000000090: AF 20 20 AE 6F 66 66 73  65 74 AF 1E 2E 28 65 78 .  .offset...(ex
0000000a0: 61 6D 70 6C 65 73 29 1E  09 11 73 70 61 74 63 68 amples)...spatch
```

```
0000000b0: 20 20 66 69 6C 65 20 20  2F 63 6D 64 73 2F 6C 73   file   /bin/ls

0000000c0: 1E 09 73 70 61 74 63 68  20 20 64 69 73 6B 20 20 ..spatch  disk
0000000d0: 31 20 20 31 1E 09 73 70  61 74 63 68 20 20 6D 65 1  1..spatch  me
0000000e0: 6D 20 20 62 30 30 30 20  20 30 10 1E 2E 28 73 74 m  b000  0...(st
0000000f0: 61 72 74 29 1E 53 50 41  54 43 48 20 69 73 20 61 art).SPATCH is a
```

At the top of the screen, you'll see a list of commands. To select a command, either type its first letter, or move the cursor to the command (with the arrow keys) and type Enter.

The commands are as follows:

Edit Enter the data area. Pressing Tab switches between hex and ASCII data entry. Pressing Esc returns you to the menu. The changed data isn't updated to the disk or memory.

Next Move forward 256 bytes. You can also use PgDn.

Prev Move backward 256 bytes. You can also use PgUp.

Home Go to the start of the file, disk, or memory. You can also use Home.

Lastblk Go to the last block of the file or disk. You can also use End.

Goto Prompt the user for an address. The type of address will depend on the source of the data (file or disk) and the address type.

Find Prompt the user for a pattern to search for. The pattern may consist of single characters or hex digit pairs separated by a space. For example:

This pattern:	Matches:
61 62 63 d e	the 5 characters "**abcde**"
a b c d e	the 5 characters "**abcde**"

© 2005, QNX Software Systems

To stop the search, press any key.

Continue Find the next occurrence of the last pattern found. Usually used after a **Find** when searching.

Save Save the current screen back to the source. Without issuing this command all changes made using **Edit** will be lost as soon as you leave the current screen of data. The **Save** option is disabled by the **-b** (browse) option.

Addr Toggle through two address types:

- Absolute (default for FILE)
- Disk Block:Offset (default for DISK)

Quit Leave the **spatch** utility.

If you specify a directory in the *file* argument, the disk will be edited, but **spatch** will move only through the blocks that make up the directory. This is similar to "spatching" a file, but if you wish to make changes, the disk must be opened for exclusive use as would any block special file.

To run **spatch** on a directory or a block special file, you must either be **root** or have write permission for the disk's block special file.

The *offset* argument lets you specify the address where **spatch** is to begin. If *file* is a regular file, the offset is a byte offset. If *file* is a block special file, the offset is a block:byte offset. If *file* is a block special file with a QNX filesystem on it, the offset may be the name of a file or directory (the beginning address will be the first block of the named file). If the named file has extents, **spatch** won't thread through those extents, but will go to the next sequential block.

You can use **spatch** to recover lost files or directories. For more information, see the QNX *Installation & Configuration* manual.

Examples:

Patch the contents of the file **/bin/ls**:

```
spatch /bin/ls
```

Patch the contents of the block-special raw disk volume **/dev/hd0**:

```
spatch /dev/hd0
```

Environment variables:

TERM Interpreted as the name of the terminal type.

Exit status:

0 Successful.

>0 An error occurred.

Caveats:

You may not use the **spatch** utility on a disk when there are open files on the disk, unless the **-b** (browse) option is used.

See also:

hd

Installation & Configuration

split

© 2005, QNX Software Systems

Split files into pieces (POSIX)

Syntax:

split [-[l] *line_count***] [-a** *suffix_length***] [***file* **[***name***]]**

split -b *n***[k|m] [-a** *suffix_length***] [***file* **[***name***]]**

Options:

-a *suffix_length*

Use *suffix_length* letters to form the suffix portion of the filenames of the split file. The default suffix length is two characters.

-b *n*[**k|m**] Split a file into pieces that are *n* bytes in size. The letters **k** or **m** may be added after *n* to specify units of kilobytes (1024 bytes) or megabytes (1048576 bytes), respectively.

-[l] *line_count*

Split the files so that there are *line_count* lines in each resulting file piece. The **l** ("el") is optional (e.g. **-50** is equivalent to **-l 50**), but the **l** should be specified since **-***nn* is an obsolescent syntax. The *line_count* argument is an unsigned decimal integer. The default is 1000. Note that the last file might differ in size from the other files.

file The pathname of the file to be split. If no files are specified or *file* is **-**, the standard input is used.

name The prefix to be used for each of the files resulting from the split operation. If *name* isn't given, **x** is used as the prefix of the output files. The combined length of *name* and *suffix_length* can't exceed 48 characters.

Description:

The **split** utility reads an input file and writes the data from that file into one or more output files.

By default, the names of the output files are **xaa**, **xab**, ..., **xzz**, and each output file, except possibly the last, gets 1000 lines.

The last file will contain the remainder of the input file, and therefore may be smaller than the requested size. Conversely, it may be longer than the other files if there are too few filenames available to take all the input in chunks of the specified size.

Examples:

Suppose you have a file named **big_file** that's 8192 lines in length. The following command will create nine files named **xaa**, **xab**, **xac**, ..., **xai**. The first eight files will all contain 1000 lines, while the last file will contain only 192.

```
split big_file
```

The following command will produce files similar to those above, except the output filenames will consist of a three-letter suffix (i.e. **xaaa**, **xaab**, **xaac**, and so on):

```
split -a 3 big_file
```

Again, assuming that **big_file** is 8192 lines in length, the following command will create only two files: **smaller_aa**, which will contain 8000 lines, and **smaller_ab**, which will contain 192 lines.

```
split -l 8000 big_file smaller_
```

Files:

You can use any file as input, but if you're splitting a non-text file, you must specify option **-b**. The output files contain portions of the original input file that are otherwise unchanged.

© 2005, QNX Software Systems

Exit status:

0	Successful completion.
>0	An error occurred.

See also:

awk, **cat**, **cut**, **head**, **sed**, **tail**

Syntax:

strings [-O *format***] [-a] [-n** *number***] [***file***...]**

Options:

-a Scan files in their entirety (default; some implementations do not scan the entire file by default — this option exists for application portability).

-n *number* Search for strings that are *number* characters or longer. The *number* argument is a positive decimal integer (default is 4 characters).

-O *format* Write each string, prefixed by its offset (in bytes). The *format* argument controls the number system used to print the offset:

To print in:	Specify:
octal	**o**
hex	**x**
decimal	**d**

The default is octal.

file The pathname of a file to be used as input. If no files are specified, **strings** reads from the standard input.

Description:

The **strings** utility looks for printable strings in regular files and writes those strings to standard output, one per line. A printable string is any sequence of *number* or more printable characters followed by a **newline** or **NUL** character.

© 2005, QNX Software Systems

Examples:

For the file **/.boot**, output all printable strings equal to or longer than four characters and prefix each line with the octal offset from the beginning of the file:

```
strings /.boot
```

Output all printable strings equal to or longer than 10 characters and prefix the line with the hex offset from the beginning of the file:

```
strings -Ox -n10 /.boot
```

Exit status:

0	Successful completion.
>0	An error occurred.

Syntax:

stty [-a|-g] [*operands***] [<** *device***]**

Options:

-a Display *all* settings.

-g Display in "getable" form.

Description:

The **stty** utility sets and/or reports terminal I/O characteristics for the device that is its standard input. If no operands are specified, **stty** displays the settings. If operands are given, then **stty** will change the terminal state to reflect those settings.

Terminal settings fall into two major categories: *edit mode* or *raw mode*. In *edit mode*, input data can be edited by the user, and is made available to programs only when an Enter is pressed. Output data will normally be presented in a human-readable format. In *raw* mode, all data flows to and from the terminal with little extra processing. If you're at a shell prompt, your terminal will probably be in the default system edit mode. A fullscreen program such as an editor, on the other hand, typically puts the terminal into raw mode.

Normally, **stty** displays only significant settings relative to the system default settings for edit or raw, and will display only the defined control characters. If **-a** or **-g** are specified, then **stty** will display *all* the settings.

The **stty** utility manages a very large number of potential terminal attributes and control characters. Most of these parameters are very device-specific and will seldom need to be changed by the user. Programs often change these terminal attributes in the course of their operation, and upon occasion (such as abnormal termination) may leave the terminal settings in an unknown state. The **stty +edit** option is a convenient method of restoring a terminal to a usable state.

Supported operands

The complete list of operands supported by **stty** are provided in the following tables. In these tables, the following conventions are used:

number represents a decimal integer number (e.g. **9600**).

name represents a string of characters (e.g. **vt100**).

value can be a single character (e.g. ~) *or* a 2-digit hexadecimal number (e.g. **1B**) *or* one of the following character pairs:

Char pair	Hex
^-	00 (undefined)
^A to ^Z	01 to 1A
^[	1B
^\	1C
^]	1D
^^	1E
^_	1F
^?	7F

An option that starts with **{+|-}** can start with either of these characters:

+ (turn the option on)

- (turn the option off)

If neither **+** nor **-** is specified, **+** is assumed.

Also note that the **=** character is optional in operands of the form *keyword***=***value*.

The following descriptions give the action taken when the option is *on*.

Line control parameters

Parameter	Defines
baud=*number*	define input and output baud rates
ispeed=*number*	define input baud rate
ospeed=*number*	define output baud rate
par=none	no parity (same as **-parenb**)
par=odd	odd parity (same as **+parenb**, **+parodd**, **-parstk**)
par=even	even parity (same as **+parenb**, **-parodd**, **-parstk**)
par=mark	mark parity (same as **+parenb**,**+parodd**, **+parstk**)
par=space	space parity (same as **+parenb**, **-parodd**, **+parstk**)
bits=5	5-bit characters
bits=6	6-bit characters
bits=7	7-bit characters
bits=8	8-bit characters
stopb=2	2-stop bits
stopb=1	1-stop bits
{+**|**-**}parenb**	enable parity
+parodd	odd parity

continued...

© 2005, QNX Software Systems

Parameter	Defines
-parodd	even parity
{+|-}parstk	stick parity
+cs5	same as **bits=5**
+cs6	same as **bits=6**
+cs7	same as **bits=7**
+cs8	same as **bits=8**
+cstopb	same as **stopb=2**
-cstopb	same as **stopb=1**
number	same as **baud=***number*
+evenp	same as **par=even**, **bits=7**
-evenp	same as **par=none**, **bits=8**
+parity	same as **par=even**, **bits=7**
-parity	same as **par=none**, **bits=8**
+oddp	same as **par=odd**, **bits=7**
-oddp	same as **par=none**, **bits=8**
{+|-}hupcl	hangup on last close
{+|-}hup	same as **hupcl**
{+|-}cread	enable receiver
{+|-}clocal	assume no modem control
{+|-}ihflow	enable hardware input flow control
{+|-}ohflow	enable hardware output flow control
{+|-}isflow	enable software input flow control
{+|-}osflow	enable software output flow control

continued...

Parameter	Defines
{+|-}lkhflow	lock hardware flow control (**ihflow**/**ohflow**)
{+|-}lksflow	lock software flow control (**isflow**/**osflow**)
{+|-}ihpaged	input is paged by hardware flow control
{+|-}ohpaged	output is paged by hardware flow control
{+|-}ispaged	input is paged by software flow control
{+|-}ospaged	output is paged by software flow control

Input processing parameters

Parameter	Defines
{+|-}ignbrk	ignore received hardware breaks
{+|-}brkint	generate SIGINT upon break
{+|-}ignpar	ignore parity errors
{+|-}parmrk	mark parity errors
{+|-}inpck	enable software parity checking
{+|-}istrip	strip 8th bit from received characters
{+|-}inlcr	map **newline** into **carriage-return** on input
{+|-}igncr	ignore received **carriage-return**
{+|-}icrnl	map **carriage-return** into **newline** on input
{+|-}ixon	same as osflow
{+|-}ixoff	same as isflow
{+|-}isig	generate signals upon receipt of special characters
{+|-}icanon	enable input line editing

continued...

Parameter	Defines
{+|-}iexten	enable “extra” special characters
{+|-}echo	echo received characters
{+|-}echoe	erase character will erase displayed character
{+|-}echok	kill character will erase displayed line
{+|-}echonl	echo **newline**, even if **ECHO** is off
{+|-}noflsh	don’t flush I/O after **INTR**, **QUIT**, or **SUSP**
+flush	flush input and output data immediately
min=*number*	minimum characters required to satisfy raw input
time=*number*	timeout value for raw input
+nl	same as **+icrnl**
-nl	same as **-icrnl**, **-inlcr**, **-igncr**
+sane	reset all parameters to sane values based on current mode (*edit*/*raw*)
+fix	same as **+sane**
+edit	reset parameters to system default *edit* mode
+raw	reset parameters to system default *raw* mode

Output processing parameters

Parameter	Defines
{+|-}opost	post-process output data

Special control characters

Parameter	Defines
eof=*value*	end-of-file character
eol=*value*	end-of-line character
erase=*value*	delete previous character
kill=*value*	delete entire line character
intr=*value*	generate SIGINT character
quit=*value*	generate SIGQUIT character
susp=*value*	generate SIGTSTP character
stop=*value*	stop output
start=*value*	resume output
+ek	resets **ERASE** and **KILL** to system defaults

Extended line-editing character sequences

QNX supports multi-character sequences to support editing capabilities in addition to the basic "erase" and "kill" functions. All of these sequences are assumed to start with an (up to) 4-character *prefix*, a single-character *action*, followed by an (up to) 4-character *suffix*. Typically, the cursor keys on your terminal will be used for these functions.

Parameter	Meaning
+load	set editing keys based on currently defined terminal type
term=*name*	set editing keys for the specified type of terminal
pr1=*value*	first character of *prefix*
pr2=*value*	second character of *prefix*

continued...

© 2005, QNX Software Systems

Parameter	Meaning
pr3=*value*	third character of *prefix*
pr4=*value*	fourth character of *prefix*
sf1=*value*	first character of *suffix*
sf2=*value*	second character of *suffix*
sf3=*value*	third character of *suffix*
sf4=*value*	fourth character of *suffix*

Action characters

The following are *action* characters when preceded by *prefix* (subsequent *suffix* will be discarded).

Parameter	Meaning
up=*value*	recall previous line
down=*value*	recall next line
left=*value*	move cursor left
right=*value*	move cursor right
ins=*value*	toggle insert mode
del=*value*	delete current character
rub=*value*	delete previous character
can=*value*	delete entire line character
home=*value*	move cursor to beginning of line
end=*value*	move cursor to end of line

Console-specific options

Parameter	Meaning
{+|-}numlock	keyboard is in Num Lock mode
{+|-}capslock	keyboard is in Caps Lock mode
{+|-}scrlock	keyboard is in Scroll Lock mode
{+|-}noboot	disable keyboard reboot
{+|-}noswitch	disable keyboard console switching
{+|-}nodebug	disable keyboard debugger entry
{+|-}noresize	disable keyboard resizing
{+|-}nohotkey	disable keyboard hotkeys
{+|-}nocolor	disable colors on console display
{+|-}monocurs	force cursor to monochrome adapter values
{+|-}extmode	put keyboard into *extended* mode
{+|-}scanmode	put keyboard into *scancode* mode
rows=*rows*[**,***cols*]	set console size
font=*value*	set console font

Serial port-specific options

Parameter	Meaning
{+|-}DTR	turn on data-terminal-ready hardware handshaking line
{+|-}RTS	turn on request-to-send hardware handshaking line
{+|-}BRK	force transmit data line to break condition

The QNX *Installation & Configuration* manual gives more detailed information concerning the circumstances under which many of these options will be required.

The **stty** utility usually displays as much information about a device as it can. Unless the **-g** option is set, the following settings may also be displayed, depending on the type of device:

Serial port information

Parameter	Meaning
{+|-}cts	clear-to-send line is on
{+|-}dsr	data-set-ready line is on
{+|-}ri	ring indicator is on
{+|-}cd	carrier detect is on

Parallel port information

Parameter	Meaning
{+|-}ERR	error condition is set
{+|-}ONL	printer is online
{+|-}PE	paper-empty indicator
{+|-}BSY	printer is busy

Examples:

Display settings of the terminal to which **stty** is attached:

```
stty
```

Display settings of a specified device:

```
stty < /dev/ser1
```

Change the baud rate of a specified device:

```
stty baud=1200 < /dev/ser1
```

Put terminal into a fixed, sane state:

```
stty +sane
```

Set editing keys to VT100 values:

```
stty term=vt100
```

Get current settings into a shell variable:

```
saveterm = "$(stty -g)"
```

Restore settings from a shell variable:

```
stty $saveterm
```

Exit status:

0 The utility executed successfully.

>0 An error occurred.

See also:

cfont

Installation & Configuration

su

© 2005, QNX Software Systems

Switch user ID (UNIX)

Syntax:

su [-c] [*userid* **[***arguments***]]**

Options:

-c Pass the specified list of arguments to the Shell.

userid Switch to the specified user ID.

Description:

The **su** utility lets a user temporarily become another user, then return to their regular *userid*.

The **su** utility requests the password of the given *userid* (**root**, by default) and changes to that *userid*, invoking its shell but modifying only essential elements of the environment. Only the **HOME**, **PATH**, and possibly **SHELL** environment variables are changed, but the new shell has the rights and privileges of the user specified. The new *userid* remains in effect until this shell exits.

Files:

/etc/passwd

This file defines the valid user IDs on the system.

See also:

login, **passwd**

sync

Update filesystems to match cached data (UNIX)

Syntax:

sync [*node*]

Options:

node (QNX extension) The node to synchronize.

Description:

The **sync** utility forces the Filesystem Manger (**Fsys**) to begin flushing all modified in-memory inodes and all previously unwritten system buffers to disk. This ensures that all file modifications up to that point are scheduled to be saved.

Pending data for all disks on the specified node, whether mounted or unmounted, will be queued for draining. If a node isn't specified, the node on which **sync** is run will be synchronized.

The **sync** utility usually returns before the operation is complete. After executing **sync**, you must allow sufficient time for the driver queues to drain before you may assume that the data is safely on disk. This delay depends on the speed of your disk(s), the number of buffers that must be drained, and how active your system is at the time. It is unusual for the operation to take more than four to five seconds with typical IDE drives and a large (2 megabyte) cache, if a large portion of the cache needs to be flushed. With faster disks (e.g. SCSI), it usually completes in under two seconds. If significant amounts of data are being flushed to a floppy disk, it could take many tens of seconds.

Examples:

Synchronize the local filesystem:

```
sync
```

Synchronize the filesystem on node 61:

© 2005, QNX Software Systems

```
sync 61
```

See also:

shutdown

syslogd

Log system messages

Syntax:

syslogd **[-f** *config_file***]** **[-m** *mark_interval***]**

Options:

-f *config_file* Specify the pathname of an alternate configuration file (the default is **/etc/syslog.conf**).

-m *mark_interval*

Select the number of minutes between "mark" messages (the default is 20 minutes).

Description:

The **syslogd** utility reads and logs messages to the system console, log files, other machines and/or users as specified by its configuration file (see below).

Whenever it receives a hangup signal, the **syslogd** utility closes its output files, reads its configuration file (default is **/etc/syslog.conf**), and then opens the output files specified by the configuration. The same process is followed at startup, except that there are no files to close.

The utility reads messages from an Internet domain socket specified in **/etc/services** and receives messages — via *Receive()* — from QNX native processes.

The name **qnx/syslog** is registered locally so that only local processes can find it (the *syslog()* function won't do a network-wide search).

To force processes to use a remote **syslogd**, the **SYSLOG** environment variable may be set to the desired node number.

The message sent to **syslogd** should consist of a single line. The message can contain a priority code, which should be a preceding decimal number in angle braces. For example:

```
<5>
```

This priority code should map into the priorities defined in the include file **sys/syslog.h**.

Configuration file

The utility reads its configuration file (**syslog.conf**) when it starts up and whenever it receives a hangup signal. The file consists of lines with two fields:

- selector field (specifies the types of messages and priorities to which the line applies)
- action field (specifies the action to be taken if a message received by **syslogd** matches the selection criteria).

The selector field is separated from the action field by one or more tab characters.

The selectors are encoded as a *facility*, a dot (**.**), and a *level*, with no intervening whitespace. Both the *facility* and the *level* are case insensitive.

The *facility* describes the part of the system generating the message, and is one of the following keywords:

- **auth**
- **authpriv**
- **cron**
- **daemon**
- **kern**
- **lpr**
- **mail**
- **mark**
- **news**

- **syslog**
- **user**
- **uucp**
- **local0** through **local7**.

These keywords (with the exception of **mark**) correspond to the similar "LOG_" values specified to the *openlog()* and *syslog()* routines.

The *level* describes the severity of the message, and is a keyword from the following ordered (highest to lowest) list:

- **emerg**
- **alert**
- **crit**
- **err**
- **warning**
- **notice**
- **info**
- **debug**

These keywords also correspond to the similar "LOG_" values specified to the *syslog()* routine.

For further descriptions of both the *facility* and *level* keywords and their significance, see the *syslog()* entry in the Watcom *C Library Reference*.

If a received message matches the specified *facility* and is of the specified (or a higher) *level*, then the action specified in the *action* field will be taken.

Multiple *selectors* may be specified for a single *action* by separating them with semicolon (**;**) characters.

Note that each *selector* can modify the ones preceding it.

Multiple facilities may be specified for a single *level* by separating them with comma (**,**) characters.

An asterisk (*****) can be used to specify all facilities or all levels.

The special *facility* **mark** receives a message at priority **info** every 20 minutes.

The special *level* **none** disables a particular *facility*.

The *action* field of each line specifies the action to be taken when the *selector* field selects a message. There are four forms:

- a pathname (beginning with a leading slash) — the selected messages are appended to the file
- a hostname (preceded by an at-sign (**@**) — the selected messages are forwarded to the **syslogd** program on the named host
- a comma-separated list of users — the selected messages are written to those users (if they're logged in)
- an asterisk — the selected messages are written to all logged-in users.

Blank lines and lines whose first non-blank character is a hash (**#**) character are ignored.

A configuration file might appear as follows:

```
# Log all kernel messages, authentication messages of
# level notice or higher and anything of level err or
# higher to the console.
# Don't log private authentication messages!
*.err;kern.*;auth.notice;authpriv.none /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none /var/log/messages

# The authpriv file has restricted access.
```

```
authpriv.* /var/log/secure

# Log all the mail messages in one place.
mail.* /var/log/maillog

# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg *
*.emerg @arpa.berkeley.edu

# Root and Eric get alert and higher messages.
*.alert root,eric

# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit /var/log/spoolerr
```


The effects of multiple selectors aren't always intuitive. For example, **mail.crit,*.err** will select **mail** facility messages at the level of **err** or higher, *not* at the level of **crit** or higher.

Files:

/etc/syslog.conf

The configuration file.

Environment variables:

SYSLOG Used by clients to specify which node to look for **syslogd**.

See also:

logger

syslog() in the Watcom *C Library Reference*

tail

© 2005, QNX Software Systems

Copy the last part of a file (POSIX)

Syntax:

tail [-c|-l] [-f] [-n *number***] [-***n***] [***file***]**

Options:

-c Measure the quantity of output in bytes.

-f If the input file is a regular file (i.e. not a tty or FIFO), don't terminate after the last line of the input file has been copied, but enter a continuous loop. The **tail** utility then sleeps for approximately one second, then attempts to read and copy further bytes from the input file.

-l ("el") Measure the quantity of output in lines; this is the default unit of measure.

-n *number* Copy output using this number of lines (default) or bytes. The *number* argument is a decimal integer whose sign specifies the location in the file to begin copying:

If the sign is:	Then copying starts relative to the:
+	beginning of the file
-	end of the file
none	end of the file

If you omit the **-n** *number* option, the default is **-10** (the number ten) i.e. the last ten units (lines or bytes) of the file.

-*n* (Obsolescent) A historical version of **-n** *number*. e.g. **-20** is equivalent to **-n 20**.

file The pathname of an input file. If you don't specify a file, the standard input is used.

Description:

The **tail** utility copies its input file to the standard output, beginning at a designated place.

Copying begins at the point in the file indicated by the **-n** *number* option. The argument *number* is counted in units of lines or bytes, according to the **-l** and **-c** options.

When **tail** is applied to a non-seekable file (e.g. a tty), **tail** must maintain an internal buffer. This buffer is large enough to hold *at least* 10 lines of characters.

Examples:

You can use the **-f** option to monitor the growth of a file that is being written by a process. For example, the command:

```
tail -f fred
```

prints the last ten lines of the file **fred**, followed by any lines that are appended to **fred** between the time **tail** is initiated and terminated. As another example, the command:

```
tail -cf -n 15 fred
```

prints the last 15 bytes of the file **fred**, followed by any lines that are appended to **fred** between the time **tail** is initiated and terminated.

Exit status:

0 Successful completion.

>0 An error occurred.

© 2005, QNX Software Systems

See also:

cat, **head**, **more**, **less**

Syntax:

```
tape [-n name] [-p n]
     [backward|bot|eod|erase|forward|
      log|position|query|reset|retension|
      rewind|status|unload|version|
      writefm]...
```

Options:

-n *name*	Use this block special device.
-p *n*	An (optional) parameter for **position**, **forward**, **backward** and **select**.
backward	Move backward to the previous file mark (EOT side).
bot	Position to the beginning of the tape.
eod	Position the tape to the end of the data (perhaps to continue writing after a previous archive).
erase	Erase the contents of the tape.
forward	Position the tape to just after the next file mark.
log	Show a log of significant events.
position	Display current tape block number.
query	Print the status of the tape driver.
reset	Reset the tape controller. This option *must* be followed by **status** to clear the exception generated by the reset.
retension	Position the tape to the beginning, then to the end, then back to the beginning.
rewind	Same as **bot**.
select	Changes tape density for next write (must be at BOM).

© 2005, QNX Software Systems

status	Print information concerning the current condition of the tape drive.
unload	Unload tape or eject cartridge.
version	Print version information for the current driver.
writefm	Write a file mark at the current position.

Description:

The **tape** utility provides a mechanism for positioning a mounted tape to a specific location in anticipation of being read from or written to by another utility (e.g. **pax**). You can also use **tape** to examine the current status of the tape unit or driver, or both.

Examples:

Position the currently mounted tape such that data may be appended after previously written archives:

```
tape eod
```

Position the tape such that the third archive on the tape may be read:

```
tape bot forward forward
```

Position the tape such that the first archive may be read:

```
tape bot
```

Exit status:

0	The command(s) completed successfully.
>0	The utility aborted after encountering the first error.

See also:

cpio, **pax**, **tar**

tar

© 2005, QNX Software Systems

Read and write tape archive files (UNIX)

Syntax:

Create a new archive:

tar -c [-b *blocking***] [-f** *file***] [-vw] [***filename***...]**

Write named files to the end of an archive:

tar -r -f *file* **[-b** *blocking***] [-vw] [***filename***...]**

List all the files contained in an archive:

tar -t [-f *file***] [-v] [***filename***]**

Extract named files from an archive:

tar -x [-f *file***] [-lmovw] [***filename***...]**

Options:

-b *blocking*	Specify the blocking factor for tape records. The default is 1. This option should be used only with raw magnetic tape archives.
-c	Create a new archive. Writing begins at the beginning of the archive, instead of after the last file.
-f *file*	Specify the name of the archive to use instead of the default, which is standard output. If the dash character (**-**) is specified as a filename, **tar** writes to the standard output or reads from the standard input, whichever is appropriate for the options given. Thus, **tar** can be used as the head or tail of a pipeline.
-l	(“el”) Report if all of the links to the files being archived cannot be resolved. If this option isn’t specified, no error messages are written to the standard output. This option is valid only with the **-c** and **-r** options.

-m Don't restore modification times. The modification time of the file will be the time of extraction. This option is invalid with the **-t** option.

-o Cause extracted files to take on the user and group identifier of the user running the program rather than those on the archive. This option is valid only with the **-x** option.

-r Write named files to the end of the archive specified in the required **-f** *file* option.

-t List the names of all of the files in the archive.

-v Be verbose. Usually, **tar** works silently, but the **-v** option causes it to print the name of each file it processes, preceded by the option letter. With the **-t** option, **-v** gives more information about the archive entries than just the name.

-w Print the action to be taken, followed by the name of the file, then wait for the user's confirmation. If you enter a word beginning with **y**, the action is performed. Any other input means "no". This option is invalid with the **-t** option.

-x Extract named files from the archive. If a named file matches a directory whose contents had been written onto the archive, that directory is recursively extracted. If a named file in the archive doesn't exist on the system, the file is created with the same mode as the one in the archive, except that the set-user-id and set-group-id modes are set only if you have appropriate privileges.

If the files exist, their modes are not changed except as described above. The owner, group, and modification time are restored if possible. If no *filename* argument is given, the entire contents of the archive is extracted. Note that if several files with the

same name are in the archive, the last one will overwrite all earlier ones.

filename The pathname of the file to be archived.

Description:

The **tar** utility reads and writes archive files that conform to the Archive/Interchange File Format specified in POSIX Std/1003.1-1988.

Examples:

Display a verbose listing of the archive members in **dist.tar**:

```
tar -tvf dist.tar
```

Copy the contents of the current directory to the floppy drive:

```
tar -cf /dev/fd0 .
```

Make an archive, **backup.tar**, of all the C source and header files in the current directory:

```
tar -cvf backup.tar *.[ch]
```

Files:

The controlling terminal (**/dev/tty**) is used to prompt the user for information when either or both the **-i** or **-y** options are specified.

See also:

cpio, **pax**, **scsi_util**, **tape**

Syntax:

tee [-ai] [*file***...]**

Options:

-a Append output to the specified file rather than overwriting it.

-i Ignore the SIGINT signal.

file A pathname of an output file. If you don't specify any files, **tee** writes to the standard output.

Description:

The **tee** utility copies standard input to standard output, making a copy in zero or more files. It does not buffer its output. You usually use **tee** in a pipeline, in order to make a copy of the output of a utility.

If **-a** is specified, the named file is opened for append and data from **tee** is added to the file's existing data. If **-a** isn't specified, the file is opened for write and the original data in the file is lost.

While it normally takes the default action for all signals, **tee** ignores SIGINT if **-i** is specified.

Examples:

Look for some spots in a collection of C programs where a variable *count* looks like it is being assigned a value, and tee the output both to standard output (where you can see it immediately) and a file **output** where you can look at it later: (note the **grep** example here is not sufficient for an exhaustive examination of where *count* might be assigned a value):

```
grep 'count *[^+-/\*]\{,1\}= *[!-([:alnum:]]' | tee output
```

© 2005, QNX Software Systems

Files:

An output file is created for each *file* operand.

Exit status:

0 Successful completion.

>0 An error occurred.

Errors:

If a write to any successfully opened file fails, writes to standard output and other successfully opened files continue. The exit status, however, will be nonzero.

Display or set the terminal type (QNX)

Syntax:

termdef [-c *command***] [-I] [-e] [***terminal_type***]**

termdef -s [*terminal_type***]**

termdef - [*terminal_type***]**

Options:

-	Write the current terminal type to the standard output. If a terminal type is specified, write the new type to the standard output.
-c *command*	After setting up the terminal type, start up *command* instead of invoking the **/bin/login -p** command.
-e	Instead of prompting for a terminal type, use the terminal type currently defined by the **TERM** environment variable. With this option, **termdef** leaves the **TERM** variable alone, but sends any necessary initialization sequences to the terminal and sets the appropriate **stty** sequences for command-line editing.
-I	Don't send terminal initialization sequences. When **-I** isn't given, **termdef** writes any terminal initialization sequences to the standard output, if required for the terminal type.
-s	Write to the standard output the Shell command for setting **TERM** to the new terminal type. If no terminal type is specified, you'll be prompted for one. If you enclose the **termdef** command in backquotes, this option will let you set the **TERM** variable without exiting the Shell (see examples).

terminal_type The terminal type to be used. If *terminal_type* is included, the user isn't prompted for a terminal type. This overrides the **-e** option.

Description:

The **termdef** utility is used for setting and querying the terminal type. It can be used from **tinit**, where it will establish the terminal type and then *exec()* into another command (usually **login**). It can also be used inside the Shell to set or query the terminal type currently being used (**termdef -** or **termdef -s**).

Using the specified terminal type (or the type currently defined by the **TERM** environment variable if **-e** is given), **termdef**:

- sets the **TERM** variable
- sends any necessary terminal initialization sequences to the terminal
- sets the **stty** sequences for command-line editing

Having configured the tty, **termdef** *exec()*s into a command that inherits the **TERM** environment variable from **termdef**. By default, this command is:

```
/bin/login -p
```

To use another command, you specify the **-c** option.

If a terminal type isn't specified or if **-e** isn't given, the user will be prompted for the terminal type. The user can obtain a list of valid terminal types by pressing Enter or by entering an invalid terminal type.

Specifying the **-e** option or a terminal type on the command line normally prevents the user from being prompted. However, if the terminal type defined by **TERM** is invalid, **termdef** displays a warning message and the user is prompted for a terminal type, as if **-e** weren't specified.

When invoked without the **-** or **-s** options, **termdef** establishes a terminal type and *exec()*s into another command (**login**, by default).

Using termdef from the Shell

You can use **termdef** from the Shell in a manner similar to the UNIX **tset** utility. To display the current terminal type, use:

```
termdef -
```

To change the terminal type, use one of the following:

This command:	Will:
termdef - *terminal_type*	write the new *terminal_type* to the standard output
termdef -s *terminal_type*	write the Shell command for setting **TERM** to the standard output
termdef -s	do the same as the above command but prompt for the terminal type

Examples:

sysinit examples in conjunction with tinit

Set up **ser2** and **ser3** to prompt for a terminal type, and *exec()* into **login** once the terminal type is established:

```
tinit -c "termdef" -T ser2 ser3 &
```

Assuming **ser1** has a VT100 terminal on it (hardwired), have **termdef** set up the tty and **TERM** variable, then *exec()* into **login**, without prompting for terminal type:

```
tinit -c "termdef vt100" -T ser1 &
```

or

© 2005, QNX Software Systems

```
tinit -c "termdef -e" -T ser1 TERM=vt100 &
```

Command-line examples

Display the terminal type:

```
termdef -
```

or

```
echo $TERM
```

Change the terminal type to VT100:

```
export TERM=‘termdef - vt100‘
```

or

```
‘termdef -s vt100‘
```

Prompt the user to specify a new terminal type:

```
‘termdef -s‘
```

Exit status:

0 **termdef** was started with valid command parameters, but may or may not have been successful. If unsuccessful, a diagnostic message will have been written to the standard error.

>0 Invalid command-line options were supplied.

See also:

infocmp, **stty**, **tic**, **tinit**

textto

© 2005, QNX Software Systems

Convert text files to QNX 2, QNX 4, or DOS format (QNX)

Syntax:

textto -c|l|r [-q] [-z] *file*...

Options:

-c Convert the specified files to DOS format (**CR/LF**).

-l ("el") Convert the specified files to QNX 4 format (**LF**).

-q Be quiet; don't write the names of files being processed to standard output.

-r Convert the specified files to QNX 2 format (**RS**).

-z Strip all ^Z codes from the file.

file The name of the file to be converted. Wildcards may be used.

Description:

The **textto** utility lets you convert text files to QNX 2, QNX 4/UNIX, or DOS format. The **-z** option can be used to strip ^Z codes from DOS-based files. Some DOS programs use this character as an end-of-file indicator.

Examples:

Change all QNX 2 record separators in **prog.c** to QNX 4 line feeds:

```
textto -l prog.c
```

Change all QNX 4 line feeds in all of the **.h** files in the current directory to QNX 2 record separators:

```
textto -r *.h
```

Change all QNX 4 line feeds in all of the **.txt** files in the current directory to DOS **CR/LF**s:

```
textto -c *.txt
```

Filter all QNX 4 line feeds in the standard input, converting them to DOS CR/LFs in the standard output:

```
textto -c - <inputfile >outputfile
```

See also:

tr

tic

© 2005, QNX Software Systems

Terminfo compiler (UNIX)

Syntax:

tic [-c] [-i] [-v *n***]** *file*

Options:

-c Perform only a syntax check of the input file. No output file is produced.

-i Ignore errors when compiling terminfo files from other systems that describe capabilities not currently supported in **/usr/lib/terminfo**.

-v *n* Set verbose output level. Progress information is output to *stderr* as **tic** works though the source file. The integer argument controls the verbosity of the report, 1 providing the least detail and 10 providing the most. The default is 0, providing "quiet" compilation.

file The pathname of a file that contains one or more terminal descriptions in **terminfo** source format (see the Terminfo Technical Note for source format details). Each description in the file describes a particular terminal. When a **use=***entry_name* field is encountered within a terminal description, **tic** reads the previously compiled description from the **/usr/lib/terminfo** directory to finish the entry. This allows terminal descriptions that are only slightly different from one previously defined to be expressed as an extension to that previous description. If the environment variable **TERMINFO** is set, the directory named will be used instead of the **/usr/lib/terminfo** directory.

Description:

The **tic** utility translates a terminfo file from the source format into the compiled format, ready to use by applications running on the terminal type named in the terminfo source file.

The compiled **terminfo** capability files are stored in single-character directories, under the **/usr/lib/terminfo** directory. For example, the VT100 terminfo file is stored in: **/usr/lib/terminfo/v/vt100**.

You can use the **infocmp** utility to convert a binary, compiled terminfo file back into a source format that can be modified and recompiled with **tic**.

Some error messages produced by **tic** will indicate the line of the source file in error and the terminal name being processed at that point.

Environment variables:

TERMINFO when defined, the compiled output from **tic** will be placed under the directory named instead of in the **/usr/lib/terminfo** directory.

Exit status:

0 The input file was compiled successfully.

>0 An error occurred.

Caveats:

The total size of a compiled terminfo description cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

Terminal names longer than 14 characters will be truncated to 14 characters.

See also:

infocmp, **terminfo** technical note (available on QUICS)

John Strang, Linda Mui, & Tim O'Reilly, *termcap & terminfo*, O'Reilly and Associates, 1988.

ticksize

© 2005, QNX Software Systems

Set the system timer granularity (QNX)

Syntax:

ticksize [-e] [*size***]**

Options:

-e Set or query ticksize in nanoseconds, using closest hardware-supported size.

size Time duration for a clock tick. If **-e** is specified, *size* is in nanoseconds, otherwise valid numbers are .5, 1, 2, 5, 10, 25, or 50 milliseconds.

Description:

The **ticksize** utility queries or changes the rate at which timer interrupts (called ticks) are applied to the system. The tick size determines the *granularity* of all software system timers. All time requests will be rounded up to this granularity. For example, if the tick size is 10 milliseconds, a request to wait for 1 millisecond may wait for up to 10 milliseconds.

Setting the tick size will not affect the system date, nor will it affect active software timers, other than the granularity of the point where they are triggered.

The timer *ticksize* is independent of the scheduling timeslice rate at which competing READY processes are scheduled to run.

Note also that QNX uses a realtime preemptive scheduler. Therefore, if a process becomes READY (perhaps as the result of an external interrupt) and is the highest priority process, it will be scheduled immediately. There is no delay based upon the tick size. Ticks are used for software timers only, not for round-robin scheduling granularity.

If your machine has a CPU speed rating of less than 600 as reported by:

```
sin info
```

then you cannot set your tick size below 1 millisecond.

When QNX boots, it sets the tick size to 10 milliseconds for machines with a CPU speed greater than or equal to 300. For machines with CPU speeds less than 300, the tick size is set to 50 milliseconds.

Examples:

Query approximate current tick size:

```
ticksize
```

Query the current tick size, to the nearest nanosecond:

```
ticksize -e
```

Set the tick size to 50 milliseconds:

```
ticksize 50
```

Set the tick size to .5 milliseconds:

```
ticksize .5
```

Exit status:

0 The tick size was successfully queried or changed.

>0 An error occurred. The tick size will not be changed.

See also:

sin

time

© 2005, QNX Software Systems

Determine the execution time of a command (UNIX)

Syntax:

time *command*

Options:

command The command, including any of its options, for which the execution time will be determined.

Description:

The **time** utility and the **time** shell built-in time the execution time of *command*, which must be specified. When execution of *command* is complete, the **time** utility displays the total elapsed time for execution, the time spent in the system, and the time spent in executing *command*. Times are reported in hours, minutes, and seconds.

Times are not always accurate to the milliseconds as displayed, depending on the tick size (see the **ticksize** utility).

Examples:

Time the execution of **dcheck -b2048 /dev/hd0**:

```
time dcheck -b2048 /dev/hd0
```

Exit status:

The **time** utility exits with the exit status of the *command* that was timed.

Caveats:

The **time** command will not obtain execution information from a remote node. To time remote programs, run **time** on the same machine as the program being timed, e.g.:

```
on -n 61 time sleep 30
```

NOT:

```
time on -n 61 sleep 30
```

See also:

date, **ticksize**

tinit

© 2005, QNX Software Systems

Terminal initialization (QNX)

Syntax:

tinit [-T *device*...**] [-c** *command***] [-f** *optionfile***]**
[-m *minutes***] [-t** *device*...**] &**

Options:

-c *command* — Start *command* on the devices specified by the **-t** or **-T** option. The default is **login -p**. This affects all **-t** and **-T** options to the right on the command line.

-f *optionfile* — The specified text file contains more command-line options. This is useful in systems with large numbers of options (typically **-c** or **-t** options) that would result in long and complex command lines.

-m *minutes* — If the line is idle (no input) for the indicated number of minutes, set a SIGHUP signal on the process group of that controlling terminal. If *minutes* is zero (the default), the idle timeouts are disabled. This affects all **-t** and **-T** options to the right on the command line.

-r *seconds* — Retry period. When **tinit** is unable to start the command on a tty (e.g. when the device is still is use by another process), it will schedule a periodic retry to see if the device has become free. The default is 60 seconds, which may be made smaller or greater using this option.

-t *device* — Start *command* immediately on this device. If *device* does not start with a **/**, it is assumed that the file exists under the **/dev** directory.

-T *device* — Start *command* on this device when any character is input. If *device* does not start with a **/**, it is assumed that the file exists under the **/dev** directory.

Description:

The **tinit** utility lets you bring up **login** (or other programs) on devices. You normally use **tinit** to invoke **login -p** on the console(s) and serial terminals. It runs as a background process, and is nearly always started in the **sysinit** file.

After creating the specified *command*s as its children, **tinit** waits for any of them to terminate. When one of these commands terminates, **tinit** re-invokes the command on that device.

For example, let's assume that **tinit** started a **login** on **/dev/con1**. After logging in, the user is presented with a shell. Then, after executing any number of commands, the user decides to terminate the shell. At this point, **tinit** detects the death and starts a new **login** on **/dev/con1**.

The list of terminal devices on which you want the command to be started can be specified by one or more **-T** or **-t** options. If a device appears more than once, the utility will use the parameters in effect for the last use on the command line. You may use a separate **-T** or **-t** for each device or use spaces in order to specify several devices, e.g.:

```
-t con2 -t con1
```

or

```
-t con2 con1
```

If the device name doesn't start with a slash, it's assumed the name starts with the following:

```
/dev/
```

The **tinit** utility scans its options left to right. Each **-c** replaces the currently defined command and each **-m** replaces the currently defined timeout. The **-t** and **-T** options memorize the current settings for starting a command on a device. For example:

```
-t con1 -c "modem -L -b 57600" -m 10 -t ser1
```

would start **login -p** (default **-c**) with no timeout (default **-m**) on **/dev/con1**, then start **modem** with a 10-minute timeout on **/dev/ser1**.

For devices specified with the **-T** option, **tinit** doesn't start the command right away. Instead, it waits for any character to be received from the device.

If you want to start several different commands on many devices, the command line for **tinit** may become very long. You can use the **-f** option to specify a file that contains options. Unlike command-line options, which are expanded by the shell, the options in this file are taken as is. For example, **-t /dev/con*** in a file will *not* be expanded.

If you want to group several blank-separated words into a single argument, enclose them in double quotes (").

Here's an example of a possible option file:

```
-t con1 -T con2 con3 con4
-c "modem -b 57600 -L" -m 10 -t ser1 ser2
-c "modem -b 57600" ser3
```

You can mix options on the command line and file options. File options are always processed *after* command-line options.

If **tinit** tries to start or restart a command, and the command fails to load, **tinit** retries once every minute until it is successful.

The **tinit** utility will log each command invocation to the file **/etc/acclog**, provided the file exists. For details, see the QNX *Installation & Configuration* manual.

Examples:

Set up **tinit** to bring up logins in response to a character being typed on any of the console devices, except **/dev/con1**, on which a login will be brought up immediately without a key being typed:

```
tinit -T /dev/con* -t con1 &
```

Set up consoles **/dev/con1** and **/dev/con2** to have **login -p** started on them immediately, and the serial port **/dev/ser1** to have **modem** started on it:

```
tinit -t con1 con2 -c "modem -b 19200 -L" -t ser1 &
```

Start **tinit** using the options specified in the file **/etc/config/tinit**:

```
tinit -f /etc/config/tinit &
```

Files:

/etc/acclog

Accounting records are appended to this file.

Caveats:

The **tinit** manager will not bring up a new login (or the specified command) regardless of **-t** or **-T** settings if any other process currently has that tty open. This means, for example, that if someone is **ditto**ing a console and logs off, nobody (neither the person running **ditto** nor the person actually sitting at the console) will be able to bring up a new **login** until **ditto** detaches from that console.

See also:

login, **modem**, **sinit**, **termdef**

touch

© 2005, QNX Software Systems

Change file access and modification times (POSIX)

Syntax:

touch [-acm] [-r *ref_file***|-t** *time***]** *file***...**

Options:

-a Change the access time of *file* to *time*, or to the current time if *time* isn't specified. Don't change the modification time unless the **-m** option is also specified.

-c If *file* doesn't already exist, don't create *file* and don't write any diagnostic messages concerning this condition.

-m Change the modification time of *file* to *time*, or to the current time if *time* isn't specified. Don't change the access time unless **-a** is also specified.

-r *ref_file* Use the specified reference file's modification time instead of the current time.

-t *time* Use the specified *time* instead of the current time, where *time* is a decimal number of the form:

[[*CC*]*YY*]*MMDDhhmm*[**.***SS*]

The two-digit pairs in *time* represent the following:

CC the first two digits of the year (i.e. the century)

YY the second two digits of the year

MM the month of the year (01-12)

DD the day of the month (01-31)

hh the hour of the day (00-23)

mm the minute of the hour (00-59)

SS the second of the minute (00-61)

Both *CC* and *YY* are optional. If you specify neither of these, the current year is assumed. If you specify *YY*, but you don't specify *CC*, *CC* is derived as follows:

If *YY* is:	*CC* becomes:
69-99	19
00-68	20

The resulting time is affected by the value of the **TZ** environment variable (see below). If the resulting time precedes 0 hours, 0 minutes, 0 seconds, January 1, 1970 Coordinated Universal Time (i.e. the Epoch), **touch** exits immediately with an error status.

The range for *SS* is 00-61 rather than 00-59 to allow leap seconds. If *SS* is 60 or 61, and the resulting time — as affected by the **TZ** environment variable — doesn't refer to a leap second, the resulting time is one second after a time where *SS* is 59. If you don't give a value to *SS*, it's assumed to be 0.

file The pathname of a file whose times are to be modified.

Description:

The **touch** utility lets you change either the modification times or access times of files, or both.

If any file you specify doesn't exist, the file is created unless you specify the **-c** option. If no time is specified, the current time is used. The **-a** option changes only the access time of the file; the **-m** option changes only the modification time of the file.

If you don't specify any options, **touch** behaves as if the **-a** and **-m** options were chosen.

© 2005, QNX Software Systems

Examples:

Set **file1**'s access and modification time to the current system time:

```
touch file1
```

Change **file2**'s access and modification time to be the same as **file1**'s modification time:

```
touch -r file1 file2
```

Set **file3**'s access and modification time to December 25, 12:34, using the current year:

```
touch -t 12251234 file3
```

Environment variables:

TZ Specifies the local time for the specified time zone.

Exit status:

0 Successful completion.

>0 An error occurred.

Syntax:

tr [-cs] [-r *filename***]** *string1 string2*

tr -d [-c] [-r *filename***]** *string1*

Options:

-c	Complement the set of characters in *string1* with respect to the universe of characters from **00** through **FF** hex. Characters in *string1* are copied unchanged while all other characters are translated.
-d	Delete all input characters in *string1* (or not in *string1* for **-dc**).
-r *filename*	(QNX extension) Translate the named file in place (don't use *stdin*/*stdout*).
-s	Squeeze all output strings of one or more instances of a single character in *string2* to a single instance of that character.
string1	Translation character string (translate from).
string2	Translation character string (translate to).

Description:

The **tr** utility copies the standard input to the standard output with substitution or deletion of selected characters. The options specified and the *string1* and *string2* operands control translations that occur while copying characters.

The default behavior is to replace each input character found in *string1* with the character at the same position in *string2*, while copying characters not in *string1* unchanged.

When *string2* is shorter than *string1*, *string2* is extended to the length of *string1* by duplicating the last character of *string2*. If *string2* is explicitly a string of zero length, it is padded with **NUL** characters.

The *string1* and *string2* operands often require quoting to avoid interpretation by the shell. Single quotes are usually the proper quoting mechanism.

Conventions for *string1* and *string2*

The following conventions can be used in *string1* or *string2* or both to specify characters, character ranges, character classes, or collating elements:

character	Represents that character.
octal	A backslash followed by 1, 2, or 3 octal digits represents a character with that encoded value.
character	A backslash followed by any character except an octal digit represents that character.
[*c***-***c***]**	Represents the range of characters between the range endpoints, inclusive.
[:*class***:]**	Represents all characters belonging to the defined character class. Allowable names for *class* are: **alpha upper lower digit xdigit alnum space punct print graph cntrl blank**
[.*cs***.]**	Represents a collating symbol. Multicharacter collating symbols must be represented as collating symbols to distinguish them from a string of the same characters. This implementation allows an arbitrary string to be treated as a collating symbol (QNX extension).
[*x*******n***]**	Represents *n* repeated occurrences of the character or collating symbol *x*. This expression is valid only in *string2*. If *n* is omitted or is zero, it is interpreted as

large enough to extend the *string2*-based sequence to the length of the *string1*-based sequence. If *n* has a leading zero, it is interpreted as an octal value. Otherwise, it is interpreted as a decimal value.

Examples:

Converts all lowercase characters in the input to the corresponding uppercase characters:

```
tr '[:lower:]' '[:upper:]' <file1 >file2
```

or

```
tr '[a-z]' '[A-Z]' <file1 >file2
```

Create a list of all words in **file1** one per line in **file2** where a word is taken to be a maximal string of letters (Octal 012 is the code for **newline**):

```
tr -cs '[:alpha:]' '[\012*]' <file 1 >file2
```

Convert a DOS file into a QNX file:

```
tr -d '\15' <infile >outfile
```

Exit status:

0 All input was processed successfully.

1 An error occurred.

See also:

awk, **sed**, **textto**

tracectrl

© 2005, QNX Software Systems

Display and change trace parameters (QNX)

Syntax:

tracectrl [-i] [-n *node***] [-p** *percent_full***] [-s** *severity***]**

Options:

-i Display current trace parameters.

-n *node* Query the specified node (default is this node).

-p *percent_full*

Set the high water mark which triggers the logging process (e.g. **tracelogger**). If you specify a *percent_full* of 0, a trigger will occur on every trace event. For more information, see **tracelogger**.

-s *severity* Tell **Proc** to save only those trace events that meet or exceed this severity level. You can specify a *severity* argument from 0 (most critical) to 7 (least critical).

Description:

The **tracectrl** utility lets you display or change trace parameters.

Trace events are stored in chronological order in the Process Manager's in-memory trace buffer. The following parameters, which **tracectrl** displays when invoked with the **-i** option, reflect the Process Manager's current trace capabilities:

Overruns The number of overruns that have occurred. An overrun occurs if the trace buffer has become completely full and another trace event is added to the buffer. Adding a trace event to the full buffer causes the oldest event to be discarded.

If overruns are occurring, you should do either or both of the following:

- increase the size of the trace buffer by specifying a larger **-T** option to **Proc** (see the documentation for **Proc**)
- consider using **tracelogger** to automatically log the trace buffer as trace events occur

Severity — A severity level. The buffer stores any events that meet or exceed this level (0 is most critical, 7 is the least).

Reader — The process ID of the logging process currently extracting events from the trace buffer. For example, if **tracelogger** is running, this value will be **tracelogger**'s pid.

BufferSize — The size, in bytes, of the trace buffer. To set this size, use **Proc**'s **-T** option.

DataSize — A running total of the number of bytes of data in the trace buffer.

Trace Selector — The memory selector of the trace buffer; **Proc** owns this selector.

High Water — The number of bytes the trace buffer must contain before the logging process is triggered. This number reflects the value entered with the **-p** option on **tracelogger** or **tracectrl**.

Proxy — The proxy to trigger when the trace buffer contains the number of bytes indicated by High Water. This proxy is set up between **Proc** and the logging process when the logging process is first run. The value has meaning only if there is a reader.

Examples:

Save events of severity 0 to 3 and trigger the logging process (e.g. **tracelogger**) when the trace buffer is 60% full:

© 2005, QNX Software Systems

```
tracectrl -s 3 -p 60
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

Proc, **traceinfo**, **tracelogger**

qnx_trace_close(), *qnx_trace_info()*, *qnx_trace_open()*, *qnx_trace_read()*, *qnx_trace_severity()*, *qnx_trace_trigger()*, and *Trace*()* functions in the Watcom C *Library Reference*.

Syntax:

traceinfo [*options***] [***logfile***...]**

Options:

-c Clear **Proc**'s trace buffer after a successful read. Trace events are cleared only if this option is specified. To use this option, you must have superuser privileges. If you specify a *logfile*, this option is ignored.

-d *start_date*[**:***end_date*]

Print trace events that occurred between *start_date* and *end_date*. If *end_date* isn't given, **traceinfo** uses the current date and time as the *end_date*.

The date format is [[[[[[*CCYY*]*MM*]*DD*]*hh*]*mm*][**.***SS*]. The utility assumes the current date and time for any omitted fields.

-e *format_file* Read additional format information from this file. The utility always reads the **/etc/config/traceinfo** file, which contains the trace format strings for the system processes (**Fsys**, **Proc**, **Net**, ...). For more info, see the section on "Trace format files."

-F *date_format_string*

Use this format string for printing the date (default is "**%b %d %T**"). This option overrides the **(date)** print variable in the trace format file. For more info, see the section on "Trace print variables."

-H *print_format_string*

Use this format string when printing trace events. This option overrides the **(header)** print variable

in the trace format file. For more info, see the section on "Trace print variables."

-m Print the millisecond timestamp after the date, according to the **(milli)** format. For more info, see the section on "Trace print variables."

-M [**!**]*major*[**:***minor*]

Print events whose major- and minor-code numbers match these values. If you don't specify a minor code, all trace events matching the major code are printed.

If you precede the major code with a **!** symbol, events with that code aren't printed.

Values may be decimal or hexadecimal (hex numbers must start with **0x**). Using multiple **-M** options, you can specify up to 10 matches and 10 exclusions.

-n *node* Get the trace events from this node (default is node on which **traceinfo** is run). This option applies only if you don't specify a logfile (i.e. trace events are being read from **Proc**'s event buffer or from standard input).

-r Print raw event data in hex.

-R Extra raw output. Trace events will be written to standard output in exactly the same format written by **tracelogger**. This is useful if you wish to store raw trace events in a file for later processing e.g. **traceinfo -R >savefile**, and later, **traceinfo savefile**.

-s [*severity1***:**]*severity2*

Print trace events whose severity level falls in the range of *severity1* to *severity2* (default is 0:7).

logfile A file containing trace events. See the **tracelogger** utility. If *logfile* is **-**, **traceinfo** reads events from standard input.

If *logfile* isn't specified, **traceinfo** reads directly from **Proc**'s trace buffer.

Description:

The **traceinfo** utility prints trace events in human-readable format. If you don't specify a logfile, **traceinfo** reads events directly from the trace buffer maintained by the Process Manager (**Proc**). If you specify one or more logfiles, **traceinfo** reads them instead of **Proc**'s trace buffer.

Events are printed one-per-line according to the format specified in the trace format file. The default format file is **/etc/config/traceinfo**. You can have **traceinfo** read additional format files by specifying the **-e** option. For more information, see the section on "Trace format files."

Trace events are binary data consisting of a *trace code* and its associated *event data*. Every trace code is made up of a 20-bit major code and a 12-bit minor code. The content of the data portion of a trace event is determined by the process that generated the event. The **traceinfo** utility simply formats and prints the event data according to print format descriptions. For example, **Proc** uses the trace code 0x00001001 for a "Bad Status" event:

Trace code	Event data
0x00001001	**03 2E 00 01 00 00 20 20 34 FF**

The trace code **0x00001001** can be further broken down into a major code and a minor code as follows:

© 2005, QNX Software Systems

Major code	Minor code
0x00001	0x001

Event data is a stream of bytes; in most cases the data is a series of integers. When a trace event is generated, **Proc** records whether the event was created by a 16- or 32-bit program. **Proc** does this because the number of bytes that constitute an integer affects how **traceinfo** processes the event data, and consequently, may affect your format specifications (integers are 2-byte items for 16-bit programs and 4-byte items for 32-bit programs).

When **traceinfo** formats the event data, it maintains a running pointer that always points to the next integer to be processed. This pointer is incremented by either 2 or 4, depending on whether the trace event was made by a 16- or 32-bit program. For example:

16-bit program	0	1	2	3	4	5
Event data	**032E**	**0001**	**0000**	**2020**	**34FF**	**22AE**
32-bit program	0		1		2	

If the above event data were made by a 16-bit program, you could print the data with the following format specification:

%d %d %d %d %d %d

Print the decimal equivalent of all 6 integers. Each integer is 2 bytes. Data pointer increments by 2 each time.

If the above event data were made by a 32-bit program, you could print the data with the following format specification:

%d %d %d Print the decimal equivalent of all 3 integers. Each integer is 4 bytes. Data pointer increments by 4 each time.

In a format string you must sometimes explicitly refer to a specific integer without using **traceinfo**'s running data pointer. To do this, you specify an *index* in the format string. An index is an integer-based offset from the beginning of the event data. Because this offset is integer-based, it will increment in the same manner as **traceinfo**'s pointer index (by either 2 or 4 bytes, depending on the program that created the event). The index starts at 0, so an index of 0 refers to the first item. For more information, see the section on "Format Specifications."

Trace format files

Trace format files describe how to print trace events by means of a *printf()*-style format specification. The structure of these files is based on special characters and command keywords.

Lines that begin with:	Indicate:
#	a *command keyword* entry
tab	a *minor-code format string*
any other characters	the line is to be ignored

Command keywords

You can specify the following command keywords:

#trace Define a trace *print variable*. For example:

```
#trace date "%b %d %y" /* use Month Day Year */
```

For more info, see the section on "Trace print variables."

#major Define a new major-code header. For example:

```
#major 0x00001 /* Process Manager */
```

All subsequent minor trace format strings refer to this major code. For more info, see "Major-code header" and "Minor-code format strings."

#table Define a new major table header. For example:

```
#table 0x00045 /* Strings for major event 0x00045 */
```

For more information, see "Tables."

#include Load another trace format file. For example:

```
#include /etc/config/screentrace
```

The specified file is opened and read as though its contents were part of the original trace format file. You can use up to five levels of nesting. For more info, see "Include files."

Trace print variables

The **traceinfo** utility has *print variables* that represent information **Proc** records internally with each event. (In the case of the event date, both a full date and a *millisecond time stamp* are recorded.) By modifying the format specification, you can control the format **traceinfo** uses when printing this information.

Note that two of the print variables aren't based on **Proc**'s internal event information:

- **(header)** — describes which of the other variables are printed and in what order.
- **(tracestr)** — the content of this variable varies depending on the event being processed. It always contains the minor-code format string for the current event being processed by **traceinfo**.

To specify print variables, use this syntax:

#trace *print_variable format_spec*

You can specify the following print variables:

(date) Use this format when printing the trace event date. This variable is overridden by the **-F** option.

String	Description	Example
%a	abbreviated weekday name	**Wed**
%A	full weekday name	**Wednesday**
%b	abbreviated month name	**Dec**
%B	full month name	**December**
%d	day of month (01-31)	**02**
%D	format *mm/dd/yy*	**12/31/97**
%H	hour as 24-hour clock	**19**
%j	day of year (001-366)	**234**
%m	month (01-12)	**04**
%T	24-hour clock *HH*:*MM*:*SS*	**23:45:12**
%r	12-hour *HH*:*MM*:*SS* **{am|pm}**	**09:45:12 PM**
%y	year (00-99)	**97**
%Y	year with century	**1997**

See also the *strftime()* function in the Watcom C *Library Reference* and the format strings supported by the **date** utility.

© 2005, QNX Software Systems

(header) Use this string to determine how other print variables are printed and in what order. This variable is overridden by the **-H** option.

Each print variable is referenced in parentheses. For example:

- Print the date of the event and the formatted event data:
  "(date) (tracestr)"
- Print major code and severity:
  "Major code=(major), Severity=(severity)"

  This would produce output similar to:
  Major code=00003, Severity=2

(major) Use this format when printing the major code of the trace event.

(milli) Use this format when printing the millisecond time stamp of the event.

(minor) Use this format when printing the minor code of the trace event.

(severity) Use this format when printing the severity of the trace event.

(tracestr) Use the minor-code format string for the current trace event. For more info, see "Minor-code format strings."

Major-code header

The command keyword **#major** defines a major-code specification that all subsequent minor-code format strings are related to.

The **#major** keyword has this syntax:

#major *major_code*

The *major_code* may be decimal or hexadecimal. If you specify it in hexadecimal, precede the number with a **0x** (e.g. **0x00001**). Anything else on the line is discarded as a comment.

The following example would start a new major-code specification for major-code 16:

```
#major  16   /* start of new major code section */
```

Minor-code format strings

These format strings follow a major-code header. They describe how to format the data for each of the minor-code events that have the preceding major code.

To specify minor-code format strings, use this syntax:

<tab>*minor_code format_spec***...**

Each line must begin with a **tab**, followed by a minor code and then a format specification. Numbers may be decimal or hexadecimal. The *format_spec* argument is a *printf()*-style string (with some extensions). For more information, see "Format specifications."

In the following example, a major-code header indicates that the subsequent minor-code format strings are for the Process Manager. The example contains two minor-code format strings (for the minor codes 10 and 12):

```
#major 0x00001     /* Process Manager */
<tab>10 Session Leader died (%s)
<tab>12 Dead loader 1 (%s)
```

Tables

Tables let you use the currently pointed-to event data as an index into a table of fixed strings. You can use this mechanism as an alternative to using minor trace codes to differentiate the various events that have the same major code. Also, if you use tables, programs don't need to put strings in their trace events, thus reducing the amount of data that must be stored in the trace buffer.

Note that table strings aren't interpreted as format strings; they're printed literally.

Each table begins with the command keyword **#table**, followed by whitespace and then a major code (which may be decimal or hexadecimal):

#table *major_code*

Here's an example:

```
#table 0x00023    /* table header for major code 0x23 */
```

Following the above keyword line, one or more table entries specify literal strings for events that can occur for the specified *major_code*. To specify these entries, use this syntax:

<tab> *index literal_string*

Each table entry uses this syntax, which consists of a tab character followed by an index number, whitespace, and then a constant string.

Here's a sample table entry:

```
#table  0x00023 /*table entries for major code 35 (0x23) */
    0   Data buffer OK.
    1   Data input buffer has overflowed!
    2   Data output buffer has overflowed!
    3   Error! Loss of carrier!
    100 Unable to continue!

#major  0x00023
    1   Error message: %(table)
```

As this example shows, you need only one minor-code format string to use the table facility.

Include files

You can include another trace format file to be read. The specified file is opened and read as if its contents were part of the original trace format file. You can have up to 5 levels of nesting.

To specify an include file, use this syntax:

#include *filename*

For example:

```
#include /etc/config/screentrace
```

Format specifications

Each format specification begins with the **%** character; the basic layout is as follows:

%[@*index***,][-][0|<space>][***min***][.***max***][S|L][***format_character***|***format_keyword***]**

where:

@*index***,**	A decimal number indicating the integer offset from the start of the event data. The index starts at 0; thus an index of 0 refers to the first integer. The index is interpreted as either a 2-byte or 4-byte integer, depending on whether the process that created the event is 16-bit or 32-bit.
-	Left justify the output.
0 *or* **<space>**	A padding character. Use a 0 to pad with zeros; use a space to pad with spaces.
[*min*]	A decimal integer indicating the minimum field width of the printed data.
[**.***max*]	A decimal integer indicating the precision of the printed data. This must be preceded by a period.

© 2005, QNX Software Systems

[**S**|**L**] A short or long indicator that will override the default integer size used when the event data is interpreted.

The default size is determined by whether the trace event was created by a 16- or 32-bit program:

S = Short (2-byte integers)
L = Long (4-byte integers)

While it affects the interpretation of formats such as **%d** and **%x**, this option does *not* affect the running pointer.

printf()-style format characters

The following format characters are supported:

Character	Description
d	decimal
u	unsigned
x	hexadecimal
b	binary
o	octal
c	char (1 byte)
s	string If used, this character must be the last field specifier in the format string. It causes any unprocessed event data to be output in ASCII without recognizing a NUL character as a string terminator.

continued...

Character	Description
e	Skip (eat) the data being pointed to. Don't print. You can use this option to move the running data pointer without printing anything.

Format keywords

Format keywords specify common ways to interpret the event data currently being pointed to. Each format keyword must be enclosed in parentheses.

(pidname) Assume the data pointer is pointing to a process ID (PID). If possible, print the process name associated with the PID.

If the PID no longer exists, or the start time of the process is later than the trace-event time, then **(pidname)** is expanded to pid=*number* or vid=*number*.

If the process name is too large to fit in the specified length, the process name starts with a * and is right justified so that the characters furthest to the right in the process name are printed.

(strerr) Assume the data pointer is pointing to an *errno* value and print that *errno*'s associated error string (see **/usr/include/errno.h**).

(errno) Assume the data pointer is pointing to an *errno* value and print that *errno*'s manifest as a string, e.g. **ENOEXEC**. (See **/usr/include/errno.h**.)

(table) Print a table string.

With this keyword, **traceinfo** looks for a **table** entry that matches the major code of the current event. If a valid table entry exists, **traceinfo** uses the value pointed to by the current data pointer as an

index into that table and prints the associated literal string.

For more information, see the section on "Tables."

(date) Assume the data pointer is pointing to a date and print the date as the output of the *ctime()* function.

(minor) Print the minor trace code associated with this trace event. The code is printed according to the format of the **(minor)** print variable.

(major) Print the major trace code associated with this trace event. The code is printed according to the format of the **(major)** print variable.

(pnet) Assume the data pointer is pointing to a physical network ID that's six bytes in length.

If the second two words of the network ID are zeros, **traceinfo** assumes the ID is an Arcnet address and prints only the first word. Otherwise the network ID is printed as a six-byte hex number.

Formatting examples

The manifests used in the following examples are defined in **/usr/include/sys/trace.h** and **/usr/include/sys/tracecod.h**.

A simple example

For the trace C code:

```
Trace2(_TRACE_TEMPORARY | 1, _TRACE_EXPECTED, 0x123, 7523);
```

the corresponding format line could be:

```
#major 0xFFFFF    /* temporary */
    1  Trace type 1 numbers are 0x%x, %d
```

and would produce:

```
Jan 19 11:34:02 Trace type 1 numbers are 0x123, 7523
```

A more advanced example

Let's now look at several format lines for the following trace C code:

```
#define _TRACE_DEMO _TRACE_TEMPORARY
/*
 * This is _TRACE_TEMPORARY until QNX Software Systems assigns
 * a trace code to your company
 */

func(pid_t sender, msg_t type) {
  msg_t    status;

  if(Send(sender, &type, sizeof(type), &status, sizeof(status)) == -1)
  {
    Trace(_TRACE_DEMO | 2, _TRACE_EXPECTED, type, sender, errno);
  } else if(status != EOK) {
    Trace(_TRACE_DEMO | 3, _TRACE_EXPECTED, type, sender, status);
  }
}
```

Here's a simple trace-format event line for this example:

```
#major 0xFFFFF    /* temporary */
    2  Send of msg %d to pid %d failed errno=%d
    3  Reply of msg %d from pid %d failed status=%d
```

Sample output:

```
Jan 19 11:34:02 Reply of msg 291 from pid 5678 failed status=22
```

You can also use *printf()*-like formats:

```
    2  Send of msg %04.4x to pid %d failed errno=%d
    3  Reply of msg %04.4x from pid %d failed status=%d
```

Sample output:

```
Jan 19 11:34:02 Reply of msg 0123 from pid 5678 failed status=22
```

You can use format keywords to expand numbers to common formats:

```
2  Send of msg %04.4x to %(pidname) failed with %(errno)
3  Reply of msg %04.4x from %(pidname) failed status=%(errno)
```

Sample output:

```
Jan 19 11:34:02 Reply of msg 0123 from /home/peterv/test
    failed with status EINVAL
```

Using an index override, you can print data that the current data pointer no longer references:

```
2  MSG=%04.4x Send failed with %@2,(errno)(%@2,d) from
       %@1,(pidname)
3  MSG=%04.4x replied with %@2,(errno)(%@2,d) from
       %@1,(pidname)
```

Sample output:

```
Jan 19 11:34:02 MSG=0123 replied with EINVAL(22) from /bin/test
```

If you also want the *errno* string printed, you could do the following:

```
2  MSG=%04.4x to %-15.15(pidname) Send  %-20.20(strerror)
       (%@2,(errno))
3  MSG=%04.4x to %-15.15(pidname) Reply %-20.20(strerror)
       (%@2,(errno))
```

Sample output:

```
Jan 19 11:34:02 MSG=0123 to /bin/test   Reply Invalid argument
    (EINVAL)
```

Passing a `long integer` from a 16-bit program

For the trace C code:

```
long a = 0x12345678;
Trace2(_TRACE_TEMPORARY | 4, _TRACE_EXPECTED, a >> 16,
     a & 0xFFFF);
```

the corresponding line could be:

```
4 Trace type 4 long number is %Ld
```

Sample output:

```
Jan 19 11:34:02 Trace type 4 long number is 305419896
```

Advanced examples

The *Trace*()* functions always pass a native **int**. If you wish to create trace events with the same information for both 16- and 32-bit programs, you may have to use overrides when printing the data. Note that you can avoid this by using different minor codes for the 16-bit and 32-bit data.

Passing a long

Let's now look at passing a long integer from a 16- or 32-bit program and interpreting this event data as a long integer.

For the trace C code:

```
long a = 0x12345678;
#ifdef __386__
Trace2(_TRACE_TEMPORARY | 5, _TRACE_EXPECTED, a);
#else
Trace2(_TRACE_TEMPORARY | 5, _TRACE_EXPECTED, a >> 16, a &
    0xFFFF);
#endif
```

the corresponding line could be:

```
5  Trace type 5 long number is %Ld
```

Sample output:

```
Jan 19 11:34:02 Trace type 5 long number is 305419896
```

Passing a short

Let's now look at passing a **short integer** from a 16- or 32-bit program and interpreting this event data as a short integer.

For the trace C code:

```
short a = 1234, b = 6789;
#ifdef __386__
Trace2(_TRACE_TEMPORARY | 6, _TRACE_EXPECTED, a | b << 16);
#else
Trace2(_TRACE_TEMPORARY | 6, _TRACE_EXPECTED, a, b);
#endif
```

the corresponding line could be:

```
6  Trace type 6 short numbers are %@0,Sd and %@1,Sd
```

Sample output:

```
Jan 19 11:34:02 Trace type 6 short numbers are 1234 and 6789
```

A complex example

The following 16-bit example generates a trace event that describes when the current machine was booted:

```
#include <stdio.h>
#include <sys/osinfo.h>
#include <sys/psinfo.h>
#include <sys/kernel.h>
#include <sys/trace.h>

char buffer[7];
struct _osinfo osdata;

#define MY_TRACE_MAJOR 0x00044  /* major trace code assigned
    by QNX Software Systems Ltd. */

main()
{
    pid_t id;
    struct _psinfo data;
    short int tmp;

    if(qnx_osinfo(getnid(), &osdata) == -1) {
        exit(-1);
        }
    id = qnx_psinfo(0, 1, &data, 0, 0); /* get psinfo for Proc */
    sprintf(buffer,"%s",
            osdata.bootsrc == 'H' ? " Hard" :
            osdata.bootsrc == 'h' ? "*Hard" :
            osdata.bootsrc == 'F' ? " Flop" :
            osdata.bootsrc == 'f' ? "*Flop" :
            osdata.bootsrc == 'N' ? "  Net" : " ????" );
    tmp = (short int)osdata.nodename;

    Trace3b( (MY_TRACE_MAJOR << 12) | 1,_TRACE_CRITICAL,
           (data.un.proc.start_time & 0x0000FFFF),  /* store
```

```
                low word first */
            (data.un.proc.start_time >> 16),
            tmp,
            strlen(buffer),
            buffer);
    }
```

This example produces a trace event similar to the following (use traceinfo **-r** to see the raw, unformatted trace output):

```
2 Aug 19 22:49:38 00043001 F792 2A92 0001 4820 7261 64
```

If you used a format string such as the following to interpret the data:

```
#major 0x00043     /* my trace code events */
    1 (%(minor)) Node %@2,d Booted on %@0,(date) : %@3,4.4s Boot
```

then the output from traceinfo would look something like this:

```
Aug 19 22:49:38 (1) Node 1 Booted on Wed Aug 19 21:38:26 1997 : Hard Boot
```

Examples:

Print trace events that have occurred since 1:00 pm today:

```
traceinfo -d 1300
```

Print trace events that have occurred between August 12 00:00 and August 13 00:00 of the current year:

```
traceinfo -d 08120000:08130000
```

Print trace events, using the given date format string:

```
traceinfo -F "%Y,%m,%d,%T"
```

Print trace events, using the given header format string:

```
traceinfo -H "(severity) (date) (tracestr)"
```

© 2005, QNX Software Systems

Print all events except for those with major code 4:

```
traceinfo -M !4
```

Print events with major code 2 and minor code 3, as well as those with major code 6:

```
traceinfo -M 2:0x03 -M 6
```

Similar to above, except that events with major code 6 and minor code 2 aren't printed:

```
traceinfo -M 2:0x03 -M 6 -M !6:2
```

Print events with severity 0 to 4:

```
traceinfo -s 4
```

Print events with severity 3:

```
traceinfo -s 3:3
```

Print events with severity 2 to 4:

```
traceinfo -s 2:4
```

Files:

/etc/config/traceinfo

/usr/include/errno.h

/usr/include/sys/trace.h

/usr/include/sys/tracecod.h

See also:

Proc, **tracectrl**, **tracelogger**

ctime(), *qnx_trace_close()*, *qnx_trace_info()*, *qnx_trace_open()*, *qnx_trace_read()*, *qnx_trace_severity()*, *qnx_trace_trigger()*, *strftime()*, and *Trace*()* functions in the Watcom C *Library Reference*.

tracelogger

© 2005, QNX Software Systems

Extract trace events and log them to a file (QNX)

Syntax:

tracelogger [*options***]**

Options:

-f *logfile* Append trace information to this logfile. The filename can include **%m** for a month expansion (01-12), **%d** for a day expansion (01-31), or both.

If you don't specify **-f**, trace events are logged to standard output, allowing you to pipe them directly into the **traceinfo** utility.

-M *mode* In conjunction with **-f**, file mode to open *logfile* with, specified as a *decimal* number. Default is 409 (O_WRONLY **|** O_CREAT **|** O_APPEND **|** O_DSYNC **|** S_IRUSR **|** S_IWUSR).

-o Optimize the writing of trace information to disk. If **-o** is used, the logfile or device specified with **-f** will be kept open as much as possible to reduce filesystem traffic.

-p *percent_full*

Extract events from the trace buffer when it becomes this full. To have events extracted as soon as they occur, set *percent_full* to 0.

If you don't specify this option, **tracelogger** will set this high water mark to 50%.

-P *permissions*

In conjunction with **-f**, permissions on the logfile, specified as a *decimal* number. Default is: 420 (S_IRUSR**|**S_IWUSR**|**S_IRGRP**|**S_IROTH)

-t *minutes* Extract events from the trace buffer if none have been extracted for this many minutes (default is 5). To disable this feature, set *minutes* to 0.

Description:

The **tracelogger** utility automatically extracts trace events and logs them to a file. The logged event data is stored in a raw binary format that you can display in human-readable form with the **traceinfo** utility. Together, these utilities provide a means of capturing runtime events that you can review when diagnosing problems.

Trace events are maintained by the Process Manager (**Proc**) in an in-memory buffer. When programs call one of the *Trace*()* library functions, a trace event is generated and quickly added to this in-memory buffer. For information on this family of functions, see the Watcom C *Library Reference*.

Most of the system managers (**Dev**, **Fsys**, **Proc**, **Net**, ...) generate trace events while they operate. To adjust the severity of the events that are logged as well as other trace parameters, use the **tracectrl** utility.

To have **tracelogger** extract data from the trace buffer immediately, you can use the SIGALRM signal. For example:

```
slay -s SIGALRM tracelogger
```

The **tracelogger** utility must be started by **root** and is typically run in the background. To stop **tracelogger**, make sure you're logged in as **root**, then type:

```
slay tracelogger
```

Examples:

Start in the background and create logfiles under **/tracedata** based on month and day (e.g. trace events on March 12 will be appended to the file **/tracedata/data.0312**). Also, extract events if the trace buffer becomes 20% full or if the 20% mark isn't reached after 3 minutes:

```
tracelogger -f /tracedata/data.%m%d -p 20 -t 3 &
```

© 2005, QNX Software Systems

Pipe trace events into the **traceinfo** program:

```
tracelogger -p 20 | traceinfo -
```

See also:

Proc, **traceinfo**, **tracectrl**

qnx_trace_close(), *qnx_trace_info()*, *qnx_trace_open()*, *qnx_trace_read()*, *qnx_trace_severity()*, *qnx_trace_trigger()*, and *Trace*()* functions in the Watcom C *Library Reference*.

true

Return true value (POSIX)

Syntax:

true

Options:

None.

Description:

The **true** utility does nothing and exits immediately with a zero exit code.

The **true** utility is typically used in shell scripts. One common application is to create an infinite loop, as in:

```
while true
do
    myprogram
    sleep 300
done
```

which runs **myprogram** every five minutes (300 seconds) until the shell script is interrupted.

Exit status:

The **true** utility always exits with a value of zero.

See also:

false

tty

© 2005, QNX Software Systems

Return user's terminal name (POSIX)

Syntax:

```
tty [-s]
```

Options:

-s Be silent; don't output the terminal name. This option is useful if you're concerned only with **tty**'s exit status.

Description:

The **tty** utility writes to the standard output the name of the terminal that is open as standard input. If standard input isn't a terminal (e.g. it is a file such as **/dev/null**), the string **not a tty** is output instead.

The **-s** option is deprecated by POSIX; the effect of this option can be achieved, simply and portably, by redirecting output to **/dev/null** or by using the Shell builtin, **test -c**.

Examples:

The following command will output **//14/dev/con1** if run on node 14, console 1.

```
tty
```

The following command will output **not a tty** since **/dev/null** causes the *isatty()* function to return 0.

```
tty </dev/null
```

Exit status:

0 Successful completion.

>0 An error occurred.

uesh

Micro-embedded shell (QNX)

Syntax:

uesh [*script_file*]

Options:

script_file A file containing shell commands to execute.

Description:

The **uesh** utility provides a subset of the functionality found in the standard QNX embedded shell, **/bin/esh**. You should find **uesh** useful for situations where memory requirements are limited. For example, you could use it to run a simple system initialization file for an embedded system.

The micro-embedded shell has some very significant limitations. These include:

- No interactive mode.
- No pipes.
- No aliases.
- No filename or command completion.
- No **set** command.

For applications that require greater functionality, use **esh** or the full shell, **sh**.

Command-line format

In **uesh**, command lines take this form:

[**//***node*] *command arg1 arg2 ...* [*redir-op file*] [**&**]

Where:

//node The node the command is to be executed on. Remote execution isn't allowed in restricted mode.

command The command to be executed. If it doesn't start with a slash, the command will follow the path set by the **PATH** environment variable. In restricted mode, a command can't start with a slash.

redir-op file A redirection operator. When a command is invoked, three standard files are set up in its environment. These files, *standard input*, *standard output*, and *standard error* output (*stdin*, *stdout*, *stderr*), are usually attached to the active terminal. You can redirect a command's standard input, standard output, and standard error as follows:

Specifying:	**Will:**
<*file*	redirect standard input from this file.
>*file*	redirect standard output to this file. If the file exists, it's overwritten; if the file doesn't exist, it's created.
>>*file*	redirect standard output to this file. If the file exists, the information is appended to the end of the file; if the file doesn't exist, it's created.
2>*file*	do the same as **>***file*, but for standard error.
2>>*file*	do the same as **>>***file*, but for standard error.

continued...

Specifying:	Will:
&	If a command contains an unquoted &, then **uesh** doesn't wait for the command to complete execution but immediately moves on to process the next command. The standard input of the command is redirected from **/dev/null**, and SIGINT and SIGQUIT are ignored.

Filename expansion

The **uesh** shell does not support filename expansion. Such shorthands as ***.c** for all files ended in **.c** will not work.

Quoting

The following characters have a special meaning in **uesh**:

```
&  \  "  *  ?  [  space
```

To suppress the special meaning of these characters and keep their literal meaning, you use quoting.

To quote a sequence of characters or sequence of words, enclose the sequence in double quotes. To quote a single character, use double quotes or precede it with the escape character (\).

Escape character (backslash)

The escape character (\) preserves the literal meaning of the next character. You can't obtain a single backslash by quoting \ with double quotes. To obtain a backslash, enter \\ instead.

Double quotes

Enclosing characters and words in double quotes (**""**) preserves the literal meaning of all characters within double quotes, with the exception of the \ character. For example:

```
"ab cd"
```

represents a single, five-character argument.

You can keep the literal meaning of a double quote with the **** character. For example:

```
ab\"cd
```

represents the single, five-character argument **ab"cd**.

Builtin commands

The following commands are built into **uesh**. That is, **uesh** interprets and executes them internally.

cd change working directory

cd [*directory*]

The **cd** utility changes the working directory of the current execution environment. If *directory* isn't specified, the value of the **HOME** environment variable becomes the new working directory. If restricted mode (**-r**) is set, you can't start the pathname with a double slash.

exec execute a command and/or manipulate file descriptors

exec [*command* [*argument*...]]

The **exec** command opens, closes, or copies file descriptors as specified by any I/O redirections given as part of *argument*. If a command is specified, that command is spawned as a replacement for **uesh**. Any specified arguments are passed to the spawned process.

exit cause **uesh** to exit

exit [*n*]

The **exit** command causes **uesh** to exit with the exit status specified by *n*. If *n* isn't specified, **uesh** exits with the status of the last command executed.

export mark environment variables for export

export *name*[**=***word*]...
export -p

The **export** command marks the specified names for **export**, which will cause them to be in the environment of subsequently executed commands. If you specify the **-p** option, the names and values of all exported variables are written to the standard output.

If restricted mode (**-r**) is set, you can't use this command.

Examples:

Invoke the script **/etc/backup** with **uesh**:

```
esh /etc/backup
```

Environment variables:

You can use the following variables to modify **uesh**'s environment:

HOME The pathname of the user's home directory

LOGNAME The user's **login** name.

PATH The directory search path used by **uesh** for locating executable programs and **uesh** shell scripts. To change **PATH**, you must use the **export** command.

If **PATH** isn't in the existing environment when **uesh** is invoked, it is set to **/bin:/usr/bin**.

SHELL The pathname of the user's preferred shell.

TERM The terminal type.

TMPDIR The pathname of a directory that can be used by utilities that need a place to create temporary files.

TZ The timezone setting.

© 2005, QNX Software Systems

See also:

esh, **sh**

umask

Get or set the file mode creation mask (POSIX)

Syntax:

umask [-o|-s|*mask***]**

Options:

-o Display the current mask, in octal.

-s Display the current mask in symbolic form. This is the default output.

mask The file mode creation *mask* can be specified either as an octal number or as a symbolic representation.

If the *mask* is specified in octal form, it replaces the current file mode creation mask. Every bit that is set describes a mode bit that *won't* be allowed in the file mode of created files. In other words, it says: "mask this bit **off**."

The symbolic form of the *mask* is an expression that modifies or replaces the current file mode creation mask. The form of the symbolic *mask* is similar to that of the mode operand for the **chmod** utility:

[[**augo**] [**+|-|=**] [**rwx**]] [**,***symbolic_mask*]

Where:

a user, group, and other access

u user access

g group access

o other access

+ add these permissions to the current mask

- remove these permissions from the current mask

= replace the current mask with these permissions

r read permission

w write permission

x execute permission

Once the symbolic mask expression has been applied to the current file mode creation mask, any occurrence of [**r**,**w**, **x**] describes a mode bit that *will* be allowed in the file mode of created files. The absence of a symbol means that permission is not allowed and is masked "off."

Description:

The **umask** utility sets the file mode creation mask of the invoking process to the value specified by the *mask* operand. The file mode creation mask affects the initial value of the file permission bits of subsequently created files when no mode is specified.

When files are created without specifying the permission mode bits, the filesystem assigns default permissions of **0777** (**rwxrwxrwx**) to directories and executable files, thereby giving read, write, and execute privileges for user, group, and others. For files that are not executable, permissions of **0666** (**rw-rw-rw-**) are assigned. The **umask** utility is used to adjust these defaults.

The file mode creation mask is inherited by any children of the current process.

Either of the display forms (**-o** or **-s**) can be used as the *mask* operand to a subsequent invocation of the **umask** utility.

As in the **chmod** utility, the use of the octal number form of mask values is deprecated.

Examples:

1 Set mask to allow read, write, and execute, for user, group, and others:

```
$ umask a=rwx
```

Display the current file creation mode mask in symbolic form:

```
$ umask -s
u=rwx, g=rwx, o=rwx
```

Display the current file creation mode mask in octal:

```
$ umask -o
00
```

2 Specify that no permissions for group and others be allowed; set the mask to allow only read and write for user only:

```
$ umask u=rw
```

Display the current file creation mode mask in symbolic form:

```
$ umask
u=rw,g=,o=
```

Display the current file creation mode mask in octal:

```
$ umask -o
0177
```

3 Add read permissions for group and others:

```
$ umask go+r
```

Display the current file creation mode mask in symbolic form:

```
$ umask
u=rw,g=r,o=r
```

Display the current file creation mode mask in octal:

```
$ umask -o
0133
```

© 2005, QNX Software Systems

Exit status:

0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.

>0 An error occurred.

Errors:

If any error occurs, the value of the process's file mode creation mask is left unmodified.

See also:

chmod

Syntax:

umount *block_special_file*

umount *mount_point*

Options:

block_special_file

The name of a block special file.

mount_point The mount point of a QNX filesystem.

Description:

The **umount** utility unmounts filesystems (i.e. mount points) or partitions mounted on the named block special file.

Once you unmount a filesystem, you can't access files on that partition — including the **mount** utility if you are getting your commands from the partition you just unmounted. Unless you have access to the **mount** utility on another disk (possibly a floppy), you will be unable to remount the filesystem.

Executing **umount** will fail if the named file isn't a mount point. It will also fail if the block special file is not mounted or if it is busy (a filesystem is busy if there are any files in use).

Since it effectively functions like **sync**, but doesn't return until all the filesystem data has been successfully updated, you should execute **umount** on floppy devices or other removable media before removing the disk (or diskette) from the drive. This will help prevent unusual errors or possible file corruption.

© 2005, QNX Software Systems

Examples:

Unmount a filesystem mounted on your primary QNX partition:

```
umount /dev/hd0t77
```

Unmount all partitions mounted on your primary hard disk:

```
umount /dev/hd0
```

Mount a read-only floppy filesystem, unmount it, and remount it as a read/write filesystem:

```
mount /dev/fd0 /floppy -r
umount /floppy
mount /dev/fd0 /floppy
```

See also:

mount, **sync**

uname

Return system name (POSIX)

Syntax:

```
uname [-amnrsv]
```

Options:

-a Behave as if all the options **-mnrsv** were specified.

-b Write the OS native word size (bits; **16** or **32**), to standard output.

-m Write the name of the hardware type on which the system is running.

-n Write the name of this node.

-r Write the current release level of the operating system (indicated by a number).

-s Write the name of the operating system.

-v Write the current version level of this release of the operating system (indicated by a letter).

Description:

The **uname** utility writes to standard output information on the name and release of the operating system being run. A portable application may use **uname** on any POSIX system to determine what operating system it's running under.

When no options are specified, **uname** writes the operating system name (QNX).

Examples:

Write the operating system name:

```
uname
```

© 2005, QNX Software Systems

Write a formatted string showing the name, release level, and version level of the operating system:

```
printf "OS: %s release %s version %s\n" `uname -srv`
```

Exit status:

0 Successful completion.

>0 An error occurred.

Caveats:

The **sin** utility provides more detailed information than **uname**, but **sin** is a QNX utility and will not be present on other systems.

See also:

sin

unexpand

Convert spaces to tabs (POSIX)

Syntax:

unexpand [-a] [-t *tabsize***] [***file***...]**

Options:

-a	In addition to the default behavior of replacing leading **space**s, translate two to eight consecutive **space**s immediately preceding a tab stop into a **tab**. A tab stop is a column position that is a multiple of eight column positions.
-t *tabsize*	(QNX extension) Set the tab stops *tabsize* columns apart (default is 8). The *tabsize* argument consists of a single positive decimal integer.
file	The pathname of a text file to be used as input.

Description:

The **unexpand** utility copies files or the standard input to the standard output, translating each group of eight **space**s at the beginning of a line into a **tab** character. Any **backspace** characters in the input are copied to the output, and each causes the column position count for tab calculations to be decremented; the count is never decremented below zero.

Examples:

For the file **sourcecode**, convert every run of eight spaces at the beginning of a line into a single tab:

```
unexpand sourcecode
```

Convert every run of two to eight spaces that precedes a tab stop into a single **tab**:

```
unexpand -a sourcecode
```

© 2005, QNX Software Systems

Convert every run of two to four spaces that precedes a tab stop into a single **tab**:

```
unexpand -a -t4 sourcecode
```

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

expand

Report or filter out repeated lines in a file (POSIX)

Syntax:

uniq [-c] [-d|-u] [-f *fields***] [-s** *chars***]**
[*input_file* **[***output_file***]]**

Deprecated syntax:

uniq [-c] [-d|-u] [-*n*] **[**+*m*] **[***input_file* **[***output_file***]]**

Options:

-*n*	(deprecated, replaced by **-f**) Ignore the first *n* fields when doing comparisons, where *n* is a number.
+*m*	(deprecated, replaced by **-s**) Ignore the first *m* characters when doing comparisons, where *m* is a number.
-c	Precede each output line with a count of the number of times the line occurred in the input.
-d	Suppress the writing of lines that aren't repeated in the input.
-f *fields*	Ignore the first *fields* on each input line when doing comparisons, where *fields* is a positive decimal integer. A field is a string of nonblank; characters separated from adjacent fields by blanks.
-s *chars*	Ignore the first *chars* characters when doing comparisons, where *chars* is a positive decimal integer. If specified in conjunction with the **-f** option, the first *chars* characters after the first *fields* fields are ignored.
-u	Suppress the writing of lines that are repeated in the input.
input_file	The pathname of the input file. If you don't specify any input files, the standard input is used.

output_file The pathname of the output file. This name must differ from the name of the input file. If you don't specify an output file, the standard output is used.

Description:

The **uniq** utility reads an input file, comparing adjacent lines, and writes one copy of each input line to the output. The second and succeeding copies of repeated adjacent input lines aren't written.

To obtain a report of unique lines in a file, the input file *must* be sorted prior to running **uniq**.

Examples:

Look for repeated adjacent lines in **datfile**:

```
uniq datfile
```

Files:

The input file is a text file.

Environment variables:

LC_TYPE The locale for character classification, used to determine the characters constituting a **blank** in the current locale.

Exit status:

0 The utility executed successfully.

>0 An error occurred.

Errors:

If *output_file* is created, it isn't removed when an error occurs.

See also:

sort

use

© 2005, QNX Software Systems

Print a usage message (QNX)

Syntax:

use [-a] *utilityname*

Options:

-a	Extract all usage information from the load module in its source form, suitable for piping into **usemsg**.
utilityname	The utility for which a usage message will be displayed.

Description:

The **use** utility prints a usage message for the specified executable program or shell script.

The **use** utility searches for *utilityname*, using the default command search (**PATH**), and prints the usage message it finds contained in the load file or shell script.

If the **LANG** environment variable is set, a usage message of that language will be displayed, if available. Alternate language usage messages aren't available within shell scripts. However, shell script messages can be easily edited. While usage messages shipped with standard versions of QNX are in English only, it is possible to add alternate language usage messages by placing the revised usage message in a separate file and using the **usemsg** utility to insert the usage message in the executable in question.

Usage messages in shell scripts

Usage messages are implemented in binary executable programs using a special form of resource record in the load modules. Usage messages are implemented in shell scripts using a format similar to that which is used in the C source code and interpreted by the **usemsg** utility.

In shell scripts, the **use** utility scans each line from the beginning of the script, looking for a line starting with the # character (i.e. a

comment) and containing the string **__USAGE**. The usage message begins on the next line and consists of all subsequent lines up to, but not including, the first line that either starts with **#endif** or starts with a character other than #.

The following is a sample usage message in a shell script:

```
#ifdef __USAGE
#%C node_number
#Where:
#  node_number   is the node you want to act on
#endif
```

If the shell script were called **foo** and you invoked **use foo** the following message would be printed:

```
foo node_number
Where:
  node_number    is the node you want to act on
```

In the above shell script fragment, the message starts with:

```
#%C node_number
```

and ends with:

```
# node_number    is the node you want to act on
```

Within the body of the usage message, the leading #s are stripped by the **use** utility and don't form part of the message that will be displayed. As with the C language usage message convention (see **usemsg**), a **%C<tab>** at the start of a line will be replaced by the program name (or filename of the shell script) and a tab character at the start of a line will space over the same number of spaces as the last previous occurrence of **%C<tab>**.

You can place the usage message almost anywhere in most shell scripts. Placing it at the beginning will result in quicker response for extracting the usage message at the expense of a very slight slowdown in execution of the shell script. If you're running a shell that doesn't recognize lines beginning with # as comments, you should place the usage message after an explicit **exit**.

© 2005, QNX Software Systems

Examples:

Print a usage message for the **ls** utility:

```
use ls
```

See also:

usemsg

Syntax:

usemsg [-c] *loadfile* **[***msgfile***]**

Options:

-c The usage message is contained in a C source program delimited by:

```
#ifdef __USAGE
...
#endif
```

Note that there are *two* underscores before **USAGE**.

loadfile The name of an executable program to extract or insert a usage message. The current **PATH** environment variable is searched to locate *loadfile*.

msgfile A text file or a C source file containing a usage message (see **-c**). If the *msgfile* name ends in **.c**, then a C source is assumed. If present, this argument will be used as the name of the message file to insert into the load file. If the *msgfile* argument is not present, the usage message is read from the load file and printed to standard output.

Description:

The **usemsg** utility lets you examine or change the usage record contained within a QNX executable program. All utilities supplied with QNX are shipped with a usage message describing their options. This information is kept in a resource record in the load file. Since this usage text is not loaded into memory when the program is run, you can make it as large as 32K characters without affecting the size of your program at runtime.

Users will use the **use** utility to print usage messages. For example:

```
use ls
use more
use sin
```

© 2005, QNX Software Systems

Developers may use the **usemsg** utility to add usage messages to their programs. There are two forms of adding a usage message to a load file. One form assumes a simple text file while the other assumes that the usage message is contained in a C source program:

```
usemsg program textfile
usemsg program program.c
```

In the second form, the C source will be scanned and all text between an **#ifdef __USAGE** and the next **#endif** will be used. In both cases any existing usage message is replaced by the new message. Note that this utility lets you both change existing usage messages and add usage messages to programs that have none. You don't need the program source.

The **usemsg** utility provides a simple grammar that allows it to support usage messages in several different languages. It also supports different messages linked to the name used to invoke the usage. For example, if **less** and **more** were links to the same load file they could each have their own usage within the same usage record within the file.

The grammar consists of the special symbol **%** in the first column followed by an action character as follows:

%%	a single %
%-*command*	the start of a specific command
%=*language*	the start of a new language
%C<tab>	replace with name of command and a space
<tab>	insert spaces equal to the length of the command + 1

To extract the entire usage message, including all languages and the grammar control sequences, you name the *loadfile* and don't specify a *msgfile*.

The **%-***command* and **%=***language* are both optional. If both are specified then the **%-***command* would be followed by one or more **%=***language*s followed by another **%-***command* and another set of **%=***language*s. The following examples should clarify the required nesting:

```
%C      a single language message

%=english
%C      an English language message
%=french
%C      a French language message

%-less
%C      single language message for less
%-more
%C      single language message for more

%-less
%=english
%C      an English language message for less
%=french
%C      a French language message for less
%-more
%=english
%C      an English language message for more
%=french
%C      a French language message for more
```

If multiple language usage messages are available, **use** employs the **LANG** environment variable to select a language. If **LANG** isn't defined or doesn't match any language present, then the first usage message is printed. Likewise, if multiple command names are present, the command passed as an argument is used to select a command. If no match occurs, the first usage message is printed.

© 2005, QNX Software Systems

Examples:

Insert usage message from C source into **myprog**:

```
usemsg myprog myprog.c
```

Extract the entire usage message for **sinit**, edit the message, then reinsert the changed message:

```
usemsg sinit >sinitmsg
vedit sinitmsg
usemsg sinit sinitmsg
```

See also:

use

Syntax:

user_smb **[-n** *node***]** **[-s** *server***]** **[-u** *userid***]**
smb-user **[***smb-password***]**

Options:

-n *node*	Tell the **SMBfsys** process on the specified node about the mapping. The default is to tell the **SMBfsys** process on the same node that **user_smb** is invoked from.
-s *server*	Specify that this mapping is valid only for the specified SMB server. The default is that the mapping applies to all servers. This allows a single QNX user ID to map to different SMB usernames on different servers. Multiple **-s** options can be specified on a single command line, in which case the mapping is set for all the named servers.
-u *userid*	The mapping is for the specified user ID number or QNX username from the **/etc/passwd** file. The default is to set the mapping for the current user. Only **root** is allowed to use this option.
smb-user	This is the SMB username on an SMB server.
smb-password	This is the SMB password for the SMB server user.

Description:

The **user_smb** utility creates a mapping between a QNX user ID and an SMB username and password. Whenever a request comes in, **SMBfsys** checks the user ID of the requesting process against the list of user ID's that it has logged in connections for the given server. If this user ID is not currently logged into the SMB server, **SMBfsys** consults the mapping tables created using **user_smb** to find out the SMB username/password and attempts to log the user in. The mappings for the particular server is checked first (**-s** option) and

then the mappings for all servers. If no mapping is found for a particular user ID, **SMBfsys** will consult the **/etc/passwd** file to find a user name. If it finds one for the user ID, it uses the given name and an empty password for the logon attempt. If **SMBfsys** can't find an entry for the user ID in **/etc/passwd**, it will use the username specified by its **-u** option and an empty password.

The **SMBfsys** manager does not store the mapping information in stable storage, so the mappings must be reestablished whenever **SMBfsys** is restarted. If **user_smb** is run with no password and the user ID/smb_user pair is new, the user will be prompted for the password. If user ID/smb_user has been seen before, the previous password for the pair is used. This allows users to place a **user_smb** command that just specifies an SMB user name in their **.profile** files. If **SMBfsys** has been restarted since the last time the user logged in to QNX, the user will be prompted to provide the password. If the information was previously entered and **SMBfsys** has not been restarted, no prompting occurs.

Examples:

Tell the **SMBfsys** manager to user the SMB ID **gates** for requests. The SMB password for **gates** is "**money**":

```
user_smb gates money
```

See also:

mount_smb, **SMBfsys**

Syntax:

vdir [*options***]... &**

Options:

-D	Dynamic (page allocator) mode. If you specify this option, **vdir** can return memory to the system when it's no longer in use.
-M	Memory allocator mode. This mode is marginally faster than dynamic mode. (**-M** is the default.)
-m *n*[**KMG**]	Maximum size of ramdisk. If you simply give a number (e.g.**-m32768**) without specifying **K**ilobytes, **M**egabytes, or **G**igabytes, the size will be in bytes. (Default is no limit.)
-n	Name of virtual directory.

Description:

The **vdir** utility lets you create a "virtual" directory that exists entirely in RAM.

This small utility is especially useful, for example, with spool files or temporary data in a memory-restricted environment. In such cases, **vdir** can be used with the **-D** option in order to increase/decrease its size as necessary.

Since **vdir** modifies the prefix tree, **root** privileges are necessary to use it.

Examples:

Create a virtual directory in "malloc" mode:

```
vdir &
```

© 2005, QNX Software Systems

Create a virtual directory named **/tmp** with a maximum size of 1 megabyte in dynamic mode:

```
vdir -D -m1M -n /tmp &
```

Caveats:

The **vdir** utility doesn't handle hard links or symbolic links.

Syntax:

vedit [*options***]...**
[*file* **[-a** *outfile1***] [-l** *nnn***]]...**

Options:

-b	Browse mode
-c *command*	Execute macro *command*
-f	Fast file save (New group and ownership)
-i *execfile*	Execute *execfile* in place of **veditrc**
-m	Use monochrome (B/W) screen colors
-o8	Enable 8 bit (graphics) chars on CRT terminal
-r	Restricted: only file(s) listed on the command line can be edited.
-s*nnn*	Restrict additional memory usage to *nnn* Kbytes
-x *execfile*	Execute autostartup file *execfile*
--	Indicates last option, permits *file* beginning with **-**
file	The pathname of a file to be edited.
-a *outfile*	The preceding *file* will be saved as *outfile*
-l *nnn*	(“el”) Editing of the preceding *file* will start on line *nnn*.

Description:

VEDIT lets you edit any files on your system, including text, program, binary, database, and mainframe files. It’s also well-suited for editing huge multi-megabyte files. You can edit in ASCII, hexadecimal, octal, or EBCDIC.

© 2005, QNX Software Systems

VEDIT supports all QNX terminals, including the main console, QNX Windows, Photon, CRT terminals, and connections via serial lines or **telnet**.

For information about the changes to vedit, see **/qnx4/vedit/whatsnew.txt**.

Getting Started Fast:

It's easiest to start VEDIT by specifying the name of the file(s) you want to edit:

vedit *filename*

You can then use the cursor keys to move around and make changes.

For help while editing, press the **[HELP]** key. The name of the help key is displayed on the status line (default: **<F1>**). **[HELP]** first displays the Table-Of-Contents for the online help. Press **[HELP]** again for help on using the online help.

The online help topics "QNX" and "QNXCFG" describe important aspects of using and configuring VEDIT that are specific to the QNX operating system.

To display the keyboard layout of the basic edit functions, select "[K]ey-layout" from the bottom line of the help viewer. For more help on any edit function, just press the corresponding key.

The **[MENU]** key (default: **<F10>**) starts the pull-down menu system. Select the desired function using the cursor keys and then press **<Enter>**.

To finish editing, access either the **{FILE}** or **{ESCAPE}** menu and select "Exit (save/abandon)". You are then prompted whether you want to save or abandon your changes.

The following pages describe VEDIT in somewhat more detail and often will refer you to particular online help topics for complete details.

Starting (Invoking) VEDIT:

Begin editing by typing the name of the VEDIT program; this is normally "**vedit**". Although not necessary, this is normally followed by the name of the file or files you want to edit or create. For example:

```
vedit letter.txt
```

```
vedit chapter1.txt chapter2.txt
```

```
vedit chapter?.txt
```

```
vedit *.c *.h
```

Invocation Options

The syntax for invoking VEDIT is:

vedit [*options***]... [***filename* **[-a** *outfile1***] [-l** *nnn***]]...**

In addition to the filename, one or more "options" can be specified when you start up VEDIT. The commonly used options are:

-b Puts VEDIT into "*browse-only*" mode; you can view files, but cannot alter them.

-m VEDIT uses the monochrome (Black/White) screen attributes. (You can configure both a set of color attributes and a set of monochrome attributes.)

-r Puts VEDIT into "*restricted*" mode; only the file(s) loaded from the command line can be edited. Additional files cannot be opened.

-- Signals the end of the options. This is needed only when the filename being edited begins with a "**-**".

filename The name of the file or files to edit. The shell will expand any wildcard characters. Each single filename may be followed by the following two options:

-a *outfile* The preceding file being edited will be saved under the name "*outfile*". This is similar to using the **{FILE, Save as}** function.

-l *nnn* ("el") Editing of the preceding file will start on line *nnn*. You can also specify a numeric expression such as "**-l(7890/3)**".

See also:

Give the commands "**use vedit**" or "**vedit -h**" for a short description of all invocation options.

The online help topic "INVOKE" fully describes all invocation options.

Exiting VEDIT:

You can selectively save or abandon each file that has been modified, or save or abandon all files at once.

To exit VEDIT and save or abandon the current file(s):

Select either **{FILE, Exit}** or **{ESCAPE, Exit}**.

If no modified files are open, this will immediately exit VEDIT. Otherwise it displays each modified file and prompts whether it is to be saved or abandoned. For example:

```
Save chapter1.txt? [Yes] [No] [Save-all] [Quit-all]
[Cancel]
```

Select "**[Yes]**" to save this file. If the file has no assigned filename, you are prompted for one.

Select "**[No]**" to abandon (quit) this file; the changes are not saved.

Select "**[Save-all]**" to save all remaining files and exit VEDIT.

Select "**[Quit-all]**" to abandon all remaining files and exit VEDIT. You may receive an additional confirmation prompt.

Notes:

When you abandon a file, you discard changes made only since the last time you saved the file (e.g. with **{FILE, Save and continue}**).

{MISC, Shell} and **{MISC, Run program}** let you execute commands for deleting files. Be careful not to delete any files that start with the same name as the file(s) you are editing. VEDIT may be using these files for temporary storage.

See also:

The online help topic "EXITING" describes exiting in more detail, including the Edit Session Restore feature and backup files.

Using VEDIT:

VEDIT is very easy and intuitive to use. New users can access most features using only a few control/functions keys, the drop-down menu system and optionally a mouse. Experienced users can speed up their editing with "hot-keys" to directly access most editing functions and bypass the menus.

The keyboard layout is fully configurable and any desired number of hot-keys (keystroke macros) can be added. The menus dynamically display the hot-keys.

Extensive context sensitive online help is always available. In particular, help is available for each dialog box.

VEDIT has over 150 configuration options including many emulation options. Common keys such as **<Enter>**, **<Tab>**, **<Home>** and

<End> can be configured to emulate almost any editor. See the online help for the **{CONFIG, Emulation}** sub-menu for more information.

Using the Keyboard

Since VEDIT's keyboard layout is fully configurable, and because you may be running VEDIT from a non-IBM type terminal, edit functions are usually referred to by name, such as **[MENU]**, instead of a particular key to press. The function name is often followed by the built-in, default key assignment, e.g. (default: **<F10>**).

To display the current (basic) keyboard layout, press the **[HELP]** key. The name of the help key is displayed on the status line (default: **<F1>**). Then press "**K**" to select "[K]ey-layout". For more help on any edit function, just press the corresponding key.

To display or print the entire keyboard layout, select the **{CONFIG, Keyboard layout}** menu and select "Display layout" or "Print layout".

To enter control characters into your file or into the search dialog box, precede it with **[ENTER CTRL]** (default: **<Ctrl-P>**).

See also:

The online help topic "KEYBOARD" explains how control and graphics characters are displayed and entered.

Dialog Boxes

VEDIT has both "full" and "terse" dialog boxes. Full dialog boxes often include more options and offer "point and shoot" selection. Terse dialog boxes are smaller and include only commonly used options.

By default, VEDIT uses full dialog boxes on high-speed QNX consoles and terse dialog boxes on slower serial connections. Erasing a terse dialog box's input field and pressing **<Enter>** switches to the full dialog box.

Press **[HELP]** (default: **<F1>**) for detailed online help on the dialog box currently displayed.

The precise operation of full and terse dialog boxes, including how the **<Tab>** key works is configurable. See the online help for the **{CONFIG, Misc}** sub-menu for more information.

See also:

The online help topic “DIALOG”.

Special Features

VEDIT has many powerful and unique editing features. Most of them are described in the numerous online help topics. To help you get started, here are the names of some of the more useful online help topics.

COLUMNS	Describes how to perform columnar block operations.
DISPLAY	Describes VEDIT’s various display modes for control and graphics characters, and how to directly edit EBCDIC files.
EBCDIC	Describes how to translate a file between ASCII, EBCDIC and ANSI. Also how to create custom translation tables.
FILETYPE	Describes how to edit DOS, UNIX/QNX and Mac type text files and convert from one type to another. Also how to edit binary and fixed-length-record (database) files.
HEX	Describes how to edit in hexadecimal and octal.
SEARCH	A very detailed description of VEDIT’s extensive search and replace capabilities. VEDIT supports both regular expressions and its own easy-to-use pattern matching language.
WORDPROC	Describes how to use the word processing features including margins, word wrap, paragraph formatting and justification.

Unique Characteristics

VEDIT is very intuitive to use, but like all programs, has a few unique characteristics which may initially confuse a new user.

- Menu Access. To save one screen line and operate faster with serial terminals, VEDIT can optionally hide the main menu under the status line at the top of the screen. The menu becomes visible when you access the menus with **[MENU]** (default: **<F10>**) or by clicking the mouse on the top status line. This feature is controlled with **{CONFIG, Screen display, Status line display}**. The presence of scrollbars can also be controlled with this menu.
- Sub-menu Preview. By default, VEDIT previews sub-menus when you navigate the menus with the cursor keys. Most other programs do not preview sub-menus. However, all programs preview sub-menus when you navigate their menus with a mouse. This feature is controlled with **{CONFIG, Misc, Enable Sub-menu preview}**. See the online help topic “MENUS” for more information.
- Persistent Block Highlighting. To make some of the block features possible, VEDIT uses persistent blocks — a block remains highlighted after you have marked both ends and move the cursor away from it. For example, you can copy/move a block directly to a new location. Therefore, you must explicitly cancel an undesired block. This is most easily done by pressing **[CANCEL]** (default: **<Ctrl-C>**). Or you can simultaneously press both mouse buttons. You can also select “Remove markers” from the **{ESCAPE}** or **{BLOCK}** menus. See the online help topic “BLOCKS” for more information.

Configuration:

Most configuration changes can be made with selections in the various **{CONFIG}**, **{PRINT, Config}** and **{SEARCH, Config}** sub-menus. While navigating these menus, press **[HELP]** (default: **<F1>**) for online help on each configuration parameter.

These configuration changes will be temporary unless you make them permanent by selecting **{CONFIG, Save to disk}**.

The keyboard layout can be changed with the **{CONFIG, Keyboard layout}** sub-menu. Similarly, selecting **{CONFIG, Keyboard layout, Save to disk}** will make any changes to the keyboard layout permanent.

The casual user does not need to know how the startup configuration works. In brief, VEDIT's configuration is controlled by the startup file "**veditrc**" in the VEDIT Home Directory, typically **/qnx4/vedit**. **veditrc** contains a series of VEDIT macro language commands that configure VEDIT to the current hardware configuration. VEDIT then further configures itself by processing one or more configuration and keyboard layout files, e.g. "**vedit.cfg**" and "**vedit.key**". Finally, VEDIT configures itself for each user by processing the "**veditrc**", "**vedit.cfg**" and "**vedit.key**" files in the user's **HOME** directory.

Experienced users may prefer to configure VEDIT by editing these files directly, but be aware that modification of "**vedit.cfg**" and "**vedit.key**" may affect the operation of VEDIT for all users.

The online help topic "QNXCFG" describes VEDIT's configuration in complete technical detail.

Printed Documentation:

Users that prefer printed documentation can print any desired portions of the online help file or can purchase the *VEDIT User's Manual* from Greenview Data, Inc.

VEDIT's online help file "**vhelp.hlp**" can easily be edited. Most of the online help topics are organized as a "User Guide" that can be printed for future reference. (It is one major chapter of the VEDIT User's Manual).

To print all of the online help topics:

1. Open the file "**vhelp.hlp**" for editing as a normal text file.
2. Search for the text "\START\".

3. Begin marking a block by selecting **{BLOCK, Set marker}** (default: **<F9>**).
4. Go to the end of the file with **{GOTO, End of file}** (default: **<Ctrl-PgDn>**).
5. Select **{PRINT, Block}** to print the highlighted block of text.
6. Select **{PRINT, Finish/Eject}** to release the print job, or simply exit VEDIT.

An extensive 380 page printed *VEDIT User's Manual* is available from Greenview Data, Inc. at a modest cost. A more powerful version of VEDIT called VEDIT PLUS is also available for QNX, Microsoft Windows, DOS and SCO UNIX. VEDIT PLUS adds integrated compiler support, color syntax highlighting for popular languages, template editing for C, HTML editing support, additional data/binary file editing capabilities, and a powerful C-like macro language which is ideal for automating editing tasks such as translations and file conversions. For additional information please contact:

```
Greenview Data, Inc., PO Box 1586, Ann Arbor, MI 48106
Toll-free: 1-800-458-3348 (US and Canada)
Telephone: 1-313-996-1300
Fax:       1-313-996-1308
E-mail:    sales@vedit.com
Web:       http://www.vedit.com
```

Normal Keyboard Layout:

This is a listing of the "normal" or default keyboard layout built into VEDIT. It includes only the basic editing functions. Select **{CONFIG, Keyboard layout, Display layout}** to view the entire layout including all keystroke macros (hot-keys).

Key Function	Key Combination
[BACKSPACE]	**<Backspace>**
[BACKTAB]	**<Shft-Tab>**
[BLOCK COPY]	**<Ctrl-F9>**
[BLOCK MOVE]	**<Alt-F9>**
[CANCEL]	**<Ctrl-C>**
[CURSOR UP]	<↑>
[CURSOR DOWN]	<↓>
[CURSOR RIGHT]	<→>
[CURSOR LEFT]	<←>
[DELETE]	**<Del>**
[DEL PREV WORD]	**<Alt-F5>**
[DEL NEXT WORD]	**<Alt-F6>**
[ENTER CTRL]	**<Ctrl-P>**
[ERASE BOL]	**<Ctrl-Bksp>**
[ERASE EOL]	**<Ctrl-Z>**
[ERASE LINE]	**<Ctrl-Y>** *or* **<Ctrl-X>**
[ESCAPE]	**<Esc>**
[HELP]	**<F1>**
[INDENT]	**<F8>**
[INSERT TOGGLE]	**<Ins>**
[LINE BEGIN]	**<Home>**
[LINE END]	**<End>**
[MENU]	**<F10>**

continued...

© 2005, QNX Software Systems

Key Function	Key Combination
[NEXT LINE]	**<Ctrl-Enter>**
[NEXT PARAGRAPH]	**<Ctrl-↓>**
[NEXT TAB STOP]	**<Ctrl-Tab>**
[NEXT WORD]	**<Ctrl-→>** *or* **<F6>**
[PAGE UP]	**<PgUp>**
[PAGE DOWN]	**<PgDn>**
[PREV PARAGRAPH]	**<Ctrl-↑>**
[PREV WORD]	**<Ctrl-←>** *or* **<F5>**
[REPEAT]	**<Ctrl-R>**
[REPEAT LAST]	**<Alt-R>**
[REPLACE]	**<Alt-F2>**
[SCREEN BEGIN]	**<Ctrl-Home>**
[SCREEN END]	**<Ctrl-End>**
[SCROLL UP]	**<Alt-↑>**
[SCROLL DOWN]	**<Alt-↓>**
[SCROLL RIGHT]	**<Alt-→>**
[SCROLL LEFT]	**<Alt-←>**
[SEARCH]	**<F2>**
[SEARCH AGAIN]	**<F3>**
[TAB CHARACTER]	**<Tab>**
[T-REG COPY]	**<Ctrl-F11>** *or* **<Grey +>**
[T-REG MOVE]	**<Alt-F11>** *or* **<Grey ->**
[T-REG INSERT]	**<F11>**

continued...

Key Function	Key Combination
[UNDENT]	**<F7>**
[VISUAL ESCAPE]	**<Alt-F10>**
[VISUAL EXIT]	**<Ctrl-F10>** *or* **<Ctrl-O>**

Examples:

Edit the file **myprog.c**:

```
vedit myprog.c
```

Edit the files **myprog.c** and **myprog.h** (two open edit buffers which you may switch between):

```
vedit myprog.c myprog.h
```

Files:

The VEDIT utility will create and/or modify the files named on the command line. In addition to these files, it will create or destroy a temporary file in the directory the file being edited is in, named

filename.*node_id*.*process_id*.*buffer_number*

When the editor is exited, these temporary files will be removed. If the editor is terminated abnormally, the temporary file(s) may remain.

In addition to the files being edited and the temporary files, **vedit** reads several configuration files, first from the VEDIT home directory (usually **/qnx4/vedit/**), then from the **$HOME** directory of the user invoking **vedit**. The main files of significance are:

$VEDIT_PATH/veditrc

Contains default **vedit** macro language commands to customize the editor to the terminal type, to the file suffix being edited, etc.

$VEDIT_PATH/vedit.cfg

default configuration information for **vedit**

$VEDIT_PATH/vedit.kbd

default keyboard layout information, in either binary or ASCII text form

$HOME/veditrc

Contains **vedit** macro language commands to customize the editor to the terminal type, to the file suffix being edited, etc.

$HOME/vedit.cfg

configuration information for **vedit**

$HOME/vedit.kbd

keyboard layout information, in either binary or ASCII text form

See the built-in help inside VEDIT under the topic "**QNXCFG**" for a complete discussion of these configuration files.

Environment variables:

HOME used as the directory to read configuration files from

TERM defines the terminal type in use

VEDIT_PATH

defines the VEDIT home directory (default **/qnx4/vedit**)

Exit status:

VEDIT always exits with a status of zero.

See also:

elvis, **qed**

vi

© 2005, QNX Software Systems

Visual display editor (UNIX)

Syntax:

vi [*options***]... [+***command***]** *file***...**

Options:

See **elvis**.

Description:

The **vi** utility is a link to **elvis**, a **vi**-compatible editor. See **elvis** for details. A matching equivalent to the **virec** utility called **elvrec** is also available. See **elvrec** for details.

See also:

ctags, **elvis**, **elvrec**, **qed**, **sed**, **vedit**

Linda Lamb, *Learning the vi Editor*, O'Reilly and Associates, 1990.

Syntax:

vol -r|-w [-b *blocking*] **[-s** *blks_num*] *block_special_device*

Options:

-b *blocking*	Use the specified blocking factor (default is 30720 bytes). A suffix of **b** or **k** may be added to the number specifying units of 512-byte blocks or kbytes, respectively.
-r	Read from the block special device.
-s *blks_num*	Skip this number of blocks at the start of each volume (default is 1).
-v	Disable delayed writes to the block special device.
-w	Write to the block special device.

Description:

You use the **vol** utility primarily with removable media, particularly floppy diskettes. At the beginning of each volume, **vol** writes a header containing the creation time and sequence number of the volume — this helps you detect out-of-sequence diskettes.

A media descriptor is located on the first block of a floppy diskette. For this reason, **vol** by default skips the first block of each volume it writes to. If you don't want it to skip the first block, you can specify the **-s** option with a value of zero. When the disk is read on a QNX system, the (overwritten) media descriptor can confuse the **Fsys.floppy** driver, which is set up to automatically switch the drive parameters for media size (360k vs 1.2M, 720k vs 1.4M etc.) based on the contents of the first block of the disk. The **lockfd** utility may be used to override this behavior and lock the floppy disk drive to a specific media type.

If you specify **-s 0**, no blocks will be skipped and no header will be written at the beginning of each volume. If you specify the **-s** option when writing an archive, remember that you must also specify the same number of blocks when the archive is retrieved.

If you want to save files larger than a single volume, specify the **-w** option. The **vol** utility will read the standard input and write it onto a block special device (usually diskettes), pausing at the start of each one.

To reassemble and extract archives split across several volumes, specify the **-r** option. The **vol** utility will read from a block special device and write the output in a single stream to standard output.

Examples:

Use **pax** to archive everything under the current directory and let **vol** split the data across as many (already-formatted) diskettes (**/dev/fd0**) as required:

```
pax -w . | vol -w /dev/fd0
```

Restore the archive from **/dev/fd0**:

```
vol -r /dev/fd0 | pax -r
```

Caveats:

If you don't tell the **vol** utility to skip zero blocks on a floppy (**-s 0**), the floppy driver might not be able to determine the media characteristics on retrieval (when the archive is written it can overwrite the media descriptor stored in the first block with data from the archive). The **lockfd** utility can be used to lock the floppy to a specific media type thus avoiding the problem.

The **vol** utility does *not* format media. See the **fdformat** utility for instructions on formatting floppy diskettes.

See also:

cpio, **lockfd**, **pax**, **tar**

wall

© 2005, QNX Software Systems

Broadcast a message (UNIX)

Syntax:

wall [-a|-l|-n *node***] [-fFTv] [***file***]**

Options:

-a Write to all users on all nodes. Counteracts **-l**. (Default)

-f Ignore the *qnx_net_alive()* node list; attempt to write to all nodes between 1 and the highest node number licensed.

-F Don't fork multiple copies. (To speed up operation, **wall** defaults to forking *num_alive_nodes*÷10 agent processes, with a maximum of 20. Each agent uses a timer and several virtual circuits.)

-l ("el") Local. Write to users on the local node only. Counteracts **-a**.

-n *node*[**-***node*]

Write to all users on the specified node, or range of nodes. A range without a second node (e.g. *n***-**) will be taken to mean "nodes *n* to the highest node ID the system is licensed for." Only a single **-n** option may be specified.

-T Test mode. All actions are performed except for the actual writing to the destination ttys. Diagnostic messages are written indicating the actions that would have been taken.

-v Verbose. Indicate persons/ttys to which messages are being delivered, as they are delivered.

-x *node*[**-***node*][**,**...]

Exclude this node from the set of nodes being written to. Node ranges may also be specified. Multiple nodes or ranges may be specified, separated by commas, e.g. **wall -x1,2,200-300** A range without a second node (e.g. *n***-**) will be taken to mean "nodes *n* to the highest node ID the system is licensed for."

file The pathname of a file containing a message to be sent (default is standard input).

Description:

You use the **wall** utility to broadcast a message to all users logged in on all nodes of the network, or optionally to users only on a specific node. By default, **wall** reads the message to be submitted from standard input. Once **wall** has been invoked, enter the message, then press Ctrl – D to submit the message.

CAUTION:

The **wall** utility now writes to all users on the network by default. Versions previous to QNX4.23 had required the **-a** option to do this. A **-l** option is available to restrict writing to users on the local node only.

If you use the **wall** utility while you are **root** (superuser), you should include your real name in the text of the message so recipients will know who is responsible for the message.

Where multiple **-a**, **-l**, or **-n** options appear, the option appearing last on the command line will take effect. Multiple **-n** node options may not be specified.

There is a ten second timeout on attempting to write to any one tty. If the ten seconds elapses before the entire message has been submitted, **wall** will abandon the effort and move on to the next tty. Usually a partial message will appear on the tty that was being written to when this occurs.

Examples:

If user **root** issues the following:

```
# wall
Warning!  Rebooting file server in three minutes for OS upgrade
If this is inconvenient come see me (glen) now!
^D
```

It will broadcast a message like this to all users on the network:

```
Broadcast message from root (//34/dev/con3) at Thu Oct 24 15:49:01
Warning!  Rebooting file server in three minutes for OS upgrade
If this is inconvenient come see me (glen) now!
EOT
```


Non-**root** users can also use **wall**, but they are not guaranteed to have write permission for the target ttys. If the user issuing **wall** does not have write permission for the tty, that tty will be skipped and **wall** will continue delivering the message to the remaining ttys.

Caveats:

If **nameloc** managers are being run on nodes which should not be providing name services (see **nameloc** for guidelines), the network alive list may show many nodes as being down which are in fact up. This can result in only a fraction of the users on the system receiving the message from **wall**. If you have users on your network who regularly break the **nameloc** rules, it is recommended that you always use the **-f** option to **wall**.

See also:

write

Syntax:

wc [-clw] [-ht] [*file***...]**

Options:

-c Write to the standard output the number of bytes in each input file.

-h (QNX extension) Display headers.

-l ("el") Write to the standard output the number of lines in each input file.

-t (QNX extension) Don't display totals.

-w Write to the standard output the number of words in each input file.

file The pathname of an input file. If no files are specified, the standard input is used.

Description:

For each input file, the **wc** utility counts lines, words, and characters (bytes) in the file and writes the results to the standard output.

If you don't specify any options, **wc** writes counts of lines, words and characters, in that order. If you specify one or more of the options **-l**, **-w**, or **-c**, the **wc** utility reports only the information selected for the specified options. The order in which you specify the options **-l**, **-w** and **-c** determines the order in which the number of lines, words, or bytes are written.

If you specify more than one input file, **wc** also writes a total count for all named files, for each option.

The utility considers a *word* to be a string of characters delimited by white space.

To get its line count, **wc** counts **newline** characters. If the last line of a file lacks the **newline** terminator, it isn't counted.

© 2005, QNX Software Systems

Files:

The input files are text files.

Exit status:

0	Successful completion.
>0	An error occurred.

which

Locate a program file (UNIX)

Syntax:

which [-aflL] *program*...

Options:

-a Find all occurrences of *program* in **PATH**.

-f Display the full QNX pathname.

-l ("el") Display long format (as in **ls -lL**) for each program found.

-L Display long format (as in **ls -l**) for each program found, displaying link information if the file is a symlink.

Description:

The **which** utility searches the filesystem for each *program* specified. The user's current **PATH** environment variable is used as the search path.

Examples:

Display full pathname (includes node) and long status for all versions of the **ls** utility found in **PATH**:

```
which -alf ls
```

Display pathname for the **which** utility:

```
which which
```

Environment variables:

PATH A colon-separated list of directories to search for QNX executables.

© 2005, QNX Software Systems

Exit status:

0	All input files were searched successfully.
>0	An error occurred.

See also:

sh

Syntax:

who [-aimow] [-n *node***]**

Options:

-a Get information on all users on the network.

-i Output the user's input "idle time" in addition to any other information. The idle time is the time since any input occurred on the user's terminal. This is the default option.

-m (me) Output information only about the current terminal.

-n *node* Get **who** information from the specified *node*. The default is the local node.

-o Output the user's output "idle time" in addition to any other information. The output idle time is the time since any output occurred to the user's terminal.

-w Show the state of each terminal in the **who** output as the **+** character if the terminal allows write access to other users and as the **-** character if such access is denied. The **?** character is printed if for some reason the information can't be determined.

Description:

The **who** utility lists various pieces of information about any accessible users.

If you don't specify any options, the default output includes the user's login name, terminal name, time at which the user logged in, input idle time, and the command being run.

If an idle time is reported as dashes, then no input (output) occurred.

© 2005, QNX Software Systems

Examples:

Display users on the current node:

```
who
```

Display information about me:

```
who -m
```

Display all users on the network:

```
who -a
```

Exit status:

0 Successful completion.

>0 An error occurred.

Caveats:

When reporting on the command being executed, the **who** utility assumes the deepest program in the process tree with the same process group number as the session leader.

See also:

login

Syntax:

write [-l] [-n *node***]** *username* **[***terminal***]**

Options:

-l	Locate the user and report active sessions (tty name and idle time) to stdout. (Does not write a message to the user.)
-n *node*	Search only the specified *node* for *username*.
username	The user to write a message to.
terminal	The terminal to write a message to.

Description:

You use the **write** utility to send a message to the terminal or console of another user.

When you invoke **write**, it writes the message:

Message from *sender* **(***sending-terminal***) [***date***]**

to the recipient's terminal. Then it copies its standard input to the recipient's terminal on a line-by-line basis until you type Ctrl – D.

The recipient can reply — thus engaging you in an interactive conversation — by entering:

write *sender sending-terminal*

If the first input character typed on a line is the exclamation mark (**!**), the **write** utility will pass the remainder of the line to the Shell (normally **sh**) for execution.

You can't send nonprintable characters through the **write** utility. They'll be expanded to octal before being written to the recipient's terminal.

When writing to a user who is logged in more than once, **write** will select one of the terminals to write to. You can use the *terminal* argument to force **write** to write to a particular terminal.

Examples:

Write to user **eric** on the local node:

```
write -n 0 eric
```

Search the network for the user **dtdodge** and write to his terminal or console:

```
write dtdodge
```

Write to **aboyd** on node 10:

```
write -n 10 aboyd
```

Write to **steve** on **//60/dev/ser1**:

```
write steve //60/dev/ser1
```

Environment variables:

SHELL Used to determine the preferred command-line interpreter for use in command escapes. If no **SHELL** environment variable is set, the **sh** utility will be used.

Exit status:

0 Successful completion.

>0 An error occurred.

See also:

wall, **who**

xargs

© 2005, QNX Software Systems

Construct argument list(s) and invoke program (POSIX)

Syntax:

xargs [-n *number***] [-s** *size***] [-t]**
[-i] [-P *n***] [***program* **[***initial-arguments***]]**

Options:

-i (QNX extension) Execute in "insert mode." The *program* will be executed *once* for each item in standard input. Each occurrence of **{}** in *initial-arguments* will be replaced with the argument read from standard input. If there are no occurrences of **{}** in *initial-arguments*, the argument is appended to the initial list.

-n *number* Every time *number* strings are read, execute *program*. The default for *number* is 255.

-P *n* (QNX extension) Use up to *n* concurrent commands. Default: 1.

-s *size* Set the maximum command buffer to *size* characters, including *program* and *initial-argument*. The default for *size* is 4096.

-t Trace. Print each *program* on standard error before executing.

program The name of the program to be executed. The program must be found by searching the path using the **PATH** environment variable. If *program* is omitted, the default is the **echo** utility.

initial-arguments

One or more arguments to *program* that are presented every time *program* is executed.

Description:

The **xargs** utility uses character strings, read from standard input, to construct a command line which it executes. The specified *program* and *initial-arguments* are placed at the beginning of the command line, followed by some number of character strings read. This process continues until the end of the file.

The utility executes a given program with *initial-arguments* one or more times using the parameters read from standard input. The number of strings appended may be limited by the **-n** option; the size of the command line may be limited by the **-s** option.

The strings are separated by **blank**s or **newline**s, which may be embedded in the strings by prefixing them with **** or enclosing them in quotes (**"**). To use the quote character as itself, you must prefix it with a ****.

The **-i** option causes the command to be executed for each string read. Instead of the normal process of appending the string to the command buffer, the *initial-arguments* are scanned, and every occurrence of **{}** is replaced by the string. If **{}** doesn't occur in *initial-arguments*, the string is appended to the command line and executed.

When a program is executed, it inherits standard output and standard error from **xargs**. Standard input is set to the controlling tty.

The **xargs** utility will always limit the total command buffer size to 4096 characters. The following example may be used to verify the integrity of data files on a floppy disk (mounted as **/fd**):

```
find /fd -print | xargs cksum | diff check_file -
```

In the example above, **find** will print out the names of each file on the mounted filesystem. The **xargs** utility will group the filenames up for **cksum** to minimize the number of times **cksum** must be executed. The **diff** utility is then used to verify that the calculated checksums are the same as recorded in the **check_file** file.

It's important to note that the following command:

```
find /fd -exec cksum {} \; | diff check_file -
```

would achieve the same thing, but would require **cksum** to be reloaded, once for each file in **/fd**.

Examples:

Use **cmp** to determine whether the files in the directory **old_data** are the same as the files in the directory **new_data**:

```
ls old_data | xargs -i cmp old_data/{} new_data/{}
```

Display the files in the current working directory, and all subdirectories, in two columns:

```
find . -print | xargs -n 2 echo
```

Exit status:

0	All invocations of *program* executed successfully.
>0	An error occurred.

See also:

find

Syntax:

yacc [-b *prefix***] [-dltv]** *file*

Options:

-b *prefix* Use *prefix* rather than **y** for the generated filename prefixes:

- **y.tab.h**
- **y.tab.c**
- **y.output**

-d Write a header file containing manifests for all tokens.

-l ("el") The file does not contain any **#***line* constructs. If this option is not specified, **yacc** will attempt to adjust any such constructs as necessary.

-t Generate a parser with debugging information, which may be enabled by setting *yydebug* to 1.

-v Be verbose; generate statistics of the parser.

Description:

The **yacc** utility transforms a specification of a context-free grammar into a C language function that implements an LR(1) parsing algorithm. Ambiguities in the grammar may be resolved using precedence rules within the specification. The C language function is named *yyparse()*, and calls *yylex()* to get input tokens, which are integer values. The value 0 is considered special, as being end-of-input.

You can use the following parser with the lexical analyzer in **lex** to implement a simple calculator.

```
%{
/*
 *   calculator:   a simple calculator.
 *   This calculator allows all common arithmetic operations,
```

```
 *   including sin, cos, tan, sqrt, exp, pow, log.
 *   The calculator is a simple example of using the yacc
 *   parser generator.
 *
 */
#include <stdio.h>
#include <math.h>

#define YYSTYPE double
    YYSTYPE last_value = 0;

extern int yylex(void);

%}

/*
  * Define the tokens produced by yylex()
 */
%token NUMBER
%token LAST
%left '+' '-'
%left '*' '/'
%left '^'
%left NEGATIVE
%left COS EXP SIN SQRT TAN

/*
 *  Begin specification of the parser.
 */
%%
/*
 * a 'list' may be empty, contain blank lines or expressions.
 */
list:
    |    list '\n'
    |    list expr '\n'           { printf("%.8g\n",last_value=$2);
}
    ; /*
 * Expressions are defined recursively as strings of terms
 * and expressions. Note that the 'sin',... functions do not
 * require bracketed parameters although sin x +1 is
 * interpreted as (sin(x))+1
 */

   expr:     term                { $$ = $1;         }
    |     expr '+' expr           { $$ = $1 + $3;    }
    |     expr '-' expr           { $$ = $1 - $3;    }
    |     expr '*' expr           { $$ = $1 * $3;    }
    |     expr '/' expr           { $$ = $1 / $3;    }
    |     expr '^' expr           { $$ = pow($1,$3); }
    |     '-' expr  %prec NEGATIVE { $$ = - $2;       }
    |     COS   term              { $$ = cos($2);    }
    |     EXP   term              { $$ = exp($2);    }
    |     SIN   term              { $$ = sin($2);    }
    |     SQRT  term              { $$ = sqrt($2);   }
    |     TAN   term              { $$ = tan($2);    }
    ;
/*
 * The following are of the highest precedence.
 * They needed to be distinguished to allow the
```

```
 * functions (sin...) to operate properly without
 * parentheses
 */
term:     NUMBER                  { $$ = $1;         }
    |     LAST                    { $$ = last_value; }
    |     '(' expr ')'            { $$ = $2;         }
    ;
%%

#include <stdlib.h>
#include <string.h>
#include <unistd.h>
int lineno;

char *fname = "-stdin-";
int yyerror(const char *s)
{
    fprintf(stderr,"%s(%d):%s\n",fname,lineno,s);
    return 0;
}
main()
{
yyparse();
return 0;
}
```

Author:

Vern Paxton, The University of California at Berkeley

See also:

lex

A.V. Aho and S.C Johnson, "LR Parsing," *Computing Surveys*, June 1974.

A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, 1977.

S.C. Johnson, "Yacc — Yet Another Compiler-Compiler," *Bell Laboratories Computing Science Technical Report # 32*, July 1978.

Brian W. Kernighan and Rob Pike, *The UNIX Programming Environment*, Prentice-Hall, 1984.

J. P. Bennett, *Introduction to Compiling Techniques — A First Course using ANSI C, LEX, and YACC*, McGraw-Hill, 1990.

© 2005, QNX Software Systems

John R. Levine, Tony Mason, and Doug Brown, *lex and yacc*, O'Reilly and Associates, 1990.

zap

Destroy a damaged file (QNX)

Syntax:

zap [-Rpv] *file*
zap [-Rpv] [-l|-u] *directory*

Options:

-R	Enter Rambo mode! Override the filesystem (allow **zap** when files are open on the disk).
-l	("el") List previously zapped files in the *directory*.
-p	Pause before starting (for floppy disks).
-u	"Unzap" files in the *directory*.
-v	Be verbose; report results.
file	The name of the file to remove.
directory	The name of the directory containing previously zapped files.

Description:

A file should be removed with **zap** if:

- the file is known to contain bad disk blocks

or

- you have been instructed to use **zap** on it by **chkfsys**

The **zap** utility releases a file by clearing the directory entry for that file. The disk blocks used by the file aren't reclaimed. Therefore, if you use **zap** repeatedly, you'll reduce the total number of disk blocks available on the disk. You can reclaim these, however, by running the **chkfsys** utility when the system is idle.

Normally, you should use the **rm** or **rmdir** commands to release files or directories.

Previously zapped files may be listed in any specified directory using the **-l** option.

You can "unzap" or recover zapped files by using the **-u** option. The utility will prompt for each file that was zapped in the specified directory.

The file to be "unzapped" *must* have been initially removed via a **zap** command. Files removed conventionally via **rm** or any other process which calls *unlink()* cannot be restored by means of the **zap -u** command.

Examples:

Eliminate the directory entry for the file **junk**:

```
zap junk
```

Exit status:

0	Successful.
>0	An error occurred.

Caveats:

To remove a file from a device that has any files open, you must use the **-R** (Rambo) option.

To run **zap** you must have read and write permission for the block special file for the filesystem containing the file being zapped. You must also have execute permission for the **zap** utility. In a normally configured system this means you must be **root** to run **zap**.

See also:

chkfsys, **rm**, **rmdir**

zcat

© 2005, QNX Software Systems

Concatenate compressed files (GNU)

Syntax:

zcat [-hLV] [*name***...]**

Options:

See **gzip**.

Description:

The **zcat** utility is part of the **gzip** suite. See **gzip** for a description.

Author:

GNU

See also:

gzip

Index

!

/etc/magic 677

B

brackets, meaning of in syntax
line 3

C

conventions
double dash, as command-line
delimiter 5
error (utilities) 7
exit status (utilities) 7
filename (utility syntax) 6
signal (utility syntax) 6
utility syntax 3
POSIX 3

D

decimal integers, in utility syntax 5
determining file type (**file**) 403
double dash, as command-line
delimiter 5
drivers
hardware 590
input 588
modules 590

E

efsinit 311
Efsys.365sl 313
Efsys.file 315
Efsys.pcmcia 317
ellipsis (**...**), meaning of in syntax
line 4
embedded filesystem (EFS)
initializing 311
event bus line 587
events 582

F

file 403
    magic number file
        (**/etc/magic**) 677
files
    determining type (**file**) 403
filesystems
    Efsys.365sl 313
    Efsys.file 315
    Efsys.pcmcia 317
Flash device
    erasing 311

G

guidelines for running utilities 4

H

hardware
    changing 592
    detecting drivers 590

I

input
    devices 589
    drivers 588
Input Manager 580–587, 589
input.*node* file 592
inputtrap 588–592
Intel
    82365 PCMCIA adapter
        filesystem 313
Intel 82540 Gigabit Ethernet network
        driver
    (**Net.i82540**) 865

M

magic 677
mouse support
    detecting 588
    protocols 580
mutually exclusive arguments 4

N

Net.sis9 878
network drivers
    SiS 900 Ethernet
        (**Net.sis9**) 878
numerical operands, in utility
        syntax 5

O

options
    multiple with single dash 4
    position of in syntax line 5
    spacing between arguments
        and 4
    syntax conventions for 3

S

SiS 900 Ethernet network driver
(**Net.sis9**) 878
starting
Input Manager 580, 589
syntax
conventions 3
principles of 3
typical example 3
utility synopsis 3

T

troubleshooting
system problems 592

U

utilities
Input Manager 580–587
inputtrap 589–592