Next Previous Contents

4. Usage and features

4.1 Tools for configuring and monitoring PCMCIA devices

If the modules are all loaded correctly, the output of the `lsmod` command should look like the following, when no cards are inserted:

```
Module                  Size  Used by
ds                      5640   2
i82365                 15452   2
pcmcia_core            30012   3  [ds i82365]
```

The system log should also include output from the socket driver describing the host controller(s) found and the number of sockets detected.

The cardmgr configuration daemon

The `cardmgr` daemon is responsible for monitoring PCMCIA sockets, loading client drivers when needed, and running user-level scripts in response to card insertions and removals. It records its actions in the system log, but also uses beeps to signal card status changes. The tones of the beeps indicate success or failure of particular configuration steps. Two high beeps indicate that a card was identified and configured successfully. A high beep followed by a low beep indicates that a card was identified, but could not be configured for some reason. One low beep indicates that a card could not be identified.

The `cardmgr` daemon configures cards based on a database of known card types kept in `/etc/pcmcia/config`. This file describes the various client drivers, then describes how to identify various cards, and which driver(s) belong with which cards. The format of this file is described in the `pcmcia(5)` **man** page.

The socket status file, stab

`Cardmgr` records device information for each socket in `/var/lib/pcmcia/stab`. Here is a sample `stab` listing:

```
Socket 0: Adaptec APA-1460 SlimSCSI
0       scsi    aha152x_cs      0       sda     8       0
0       scsi    aha152x_cs      1       scd0    11      0
Socket 1: Serial or Modem Card
1       serial  serial_cs       0       ttyS1   5       65
```

For the lines describing devices, the first field is the socket, the second is the device class, the third is the driver name, the fourth is used to number multiple devices associated with the same driver, the fifth is the device name, and the final two fields are the major and minor device numbers for this device (if applicable). See the `stab` **man** page for more info.

In 2.4 and later kernels, hot plut PCI drivers for CardBus cards are not managed by `cardmgr`; they are managed by the `hotplug` subsystem. See http://linux-hotplug.sourceforge.net for information about this facility. When `cardmgr` sees a card that is owned by a hot plug PCI driver, it will ignore that card. There will be one beep when these cards are inserted or ejected, but they will be identified only as a ``CardBus hotplug device'' in the system log and `stab` file.

The cardctl and cardinfo utilities

The `cardctl` command can be used to check the status of a socket, or to see how it is configured. It can also be used to alter the configuration status of a card. Here is an example of the output of the ``cardctl config'' command:

```
Socket 0:
  not configured
Socket 1:
  Vcc = 5.0, Vpp1 = 0.0, Vpp2 = 0.0
  Card type is memory and I/O
  IRQ 3 is dynamic shared, level mode, enabled
  Speaker output is enabled
  Function 0:
    Config register base = 0x0800
      Option = 0x63, status = 0x08
    I/O window 1: 0x0280 to 0x02bf, auto sized
    I/O window 2: 0x02f8 to 0x02ff, 8 bit
```

Or ``cardctl ident'', to get card identification information:

```
Socket 0:
  no product info available
Socket 1:
  product info: "LINKSYS", "PCMLM336", "A", "0040052D6400"
  manfid: 0x0143, 0xc0ab
  function: 0 (multifunction)
```

The ``cardctl suspend'' and ``cardctl resume'' commands can be used to shut down a card without unloading its associated drivers. The ``cardctl reset'' command attempts to reset and reconfigure a card. ``cardctl insert'' and ``cardctl eject'' mimic the actions performed when a card is physically inserted or ejected, including loading or unloading drivers, and configuring or shutting down devices.

If you are running X, the `cardinfo` utility produces a graphical display showing the current status of all PCMCIA sockets, similar in content to ``cardctl config''. It also provides a graphical interface to most other `cardctl` functions.

Inserting and ejecting cards

In theory, you can insert and remove PCMCIA cards at any time. However, it is a good idea not to eject a card that is currently being used by an application program. Kernels older than 1.1.77 would often lock up when serial/modem cards were ejected, but this should be fixed now.

Some card types cannot be safely hot ejected. Specifically, ATA/IDE and SCSI interface cards are not hot-swap-safe. This is unlikely to be fixed, because a complete solution would require significant changes to the Linux block device model. Also, it is generally not safe to hot eject CardBus cards of any type. This is likely to improve gradually as hot swap bugs in the CardBus drivers are found and fixed. For these card types (IDE, SCSI, CardBus), it is recommended that you always use ``cardctl eject'' before ejecting.

Card Services and Advanced Power Management

Card Services can be compiled with support for APM (Advanced Power Management) if you've configured your kernel with APM support. The APM kernel driver is maintained by Stephen Rothwell (Stephen.Rothwell@canb.auug.org.au). The `apmd` daemon is maintained by Avery Pennarun (apenwarr@worldvisions.ca), with more information available at http://www.worldvisions.ca/~apenwarr/apmd/. The PCMCIA modules will automatically be configured for APM if a compatible version is detected on your system.

Whether or not APM is configured, you can use ``cardctl suspend'' before suspending your laptop, and ``cardctl resume'' after resuming, to cleanly shut down and restart your PCMCIA cards. This will not work with a modem that is in use, because the serial driver isn't able to save and restore the modem operating parameters.

APM seems to be unstable on some systems. If you experience trouble with APM and PCMCIA on your system, try to narrow down the problem to one package or the other before reporting a bug.

Some drivers, notably the PCMCIA SCSI drivers, cannot recover from a suspend/resume cycle. When using a PCMCIA SCSI card, always use ``cardctl eject'' prior to suspending the system.

Shutting down the PCMCIA system

To unload the entire PCMCIA package, invoke `rc.pcmcia` with:

```
/etc/rc.d/rc.pcmcia stop
```

This script will take several seconds to run, to give all client drivers time to shut down gracefully. If a device is currently in use, the shutdown will be incomplete, and some kernel modules may not be unloaded. To avoid this, use ``cardctl eject'' to shut down all sockets before invoking `rc.pcmcia`. The exit status of the `cardctl` command will indicate if any sockets could not be shut down.

4.2 Overview of the PCMCIA configuration scripts

The following information applies to cards that are managed by `cardmgr`. In 2.4 and later kernels, if the kernel PCMCIA subsystem is active, then CardBus cards are managed by the `hotplug` subsystem and the PCMCIA scripts are not used.

Each PCMCIA device has an associated ``class'' that describes how it should be configured and managed. Classes are associated with device drivers in `/etc/pcmcia/config`. There are currently five IO device classes (network, SCSI, cdrom, fixed disk, and serial) and two memory device classes (memory and FTL). For each class, there are two scripts in `/etc/pcmcia`: a main configuration script (i.e., `/etc/pcmcia/scsi` for SCSI devices), and an options script (i.e., `/etc/pcmcia/scsi.opts`). The main script for a device will be invoked to configure that device when a card is inserted, and to shut down the device when the card is removed. For cards with several associated devices, the script will be invoked for each device.

The config scripts start by extracting some information about a device from the `stab` file. Each script constructs a ``device address'', that uniquely describes the device it has been asked to configure, in the `ADDRESS` shell variable. This is passed to the `*.opts` script, which should return information about how a device at this address should be configured. For some devices, the device address is just the socket number. For others, it includes extra information that may be useful in deciding how to configure the device. For example, network devices pass their hardware ethernet address as part of the device address, so the `network.opts` script could use this to select from several different configurations.

The first part of all device addresses is the current PCMCIA ``scheme''. This parameter is used to support multiple sets of device configurations based on a single external user-specified variable. One use of schemes would be to have a ``home'' scheme, and a ``work'' scheme, which would include different sets of network configuration parameters. The current scheme is selected using the ```cardctl scheme`'' command. The default if no scheme is set is ``default''.

There are a few additional shell variables that can be used in `*.opts` files in addition to `ADDRESS`:

`SOCKET`, `CLASS`, `DRIVER`, `INSTANCE`, `DEVICE`, `MAJOR`, `MINOR`

> These correspond to individual fields from one line in the `stab` file. See its **man** page for details.

`PRODID_1`, `PRODID_2`, `PRODID_3`, `PRODID_4`, `MANFID`, `FUNCID`

> These are equivalent to the output of ```cardctl info`'' and give more detailed card identification information.

As the `*.opts` files are just shell scripts, it is not required that they follow the form of the examples, which just return settings based on `ADDRESS`.

As a general rule, when configuring Linux for a laptop, PCMCIA devices should only be configured from the PCMCIA device scripts. Do not try to configure a PCMCIA device the same way you would configure a permanently attached device. However, some Linux distributions provide PCMCIA packages that are hooked into those distributions' own device configuration tools. In that case, some of the following sections may not apply; ideally, this will be documented by the distribution maintainers.

4.3 PCMCIA network adapters

Linux ethernet-type network interfaces normally have names like `eth0`, `eth1`, and so on. Token-ring adapters are handled similarly, however they are named `tr0`, `tr1`, and so on. The `ifconfig` command is used to view or modify the state of a network interface. A peculiarity of Linux is that network interfaces do not have corresponding device files under `/dev`, so do not be surprised when you do not find them.

When an ethernet card is detected, it will be assigned the first free interface name, which will normally be `eth0`. `Cardmgr` will run the `/etc/pcmcia/network` script to configure the interface, which normally reads network settings from `/etc/pcmcia/network.opts`. The `network` and `network.opts` scripts will be executed only when your ethernet card is actually present. If your system has an automatic network configuration facility, it may or may not be PCMCIA-aware. Consult the documentation of your Linux distribution and the Notes about specific Linux distributions to determine if PCMCIA network devices should be configured with the automatic tools, or by editing `network.opts`.

The device address passed to `network.opts` consists of four comma-separated fields: the scheme, the socket number, the device instance, and the card's hardware ethernet address. The device instance is used to number devices for cards that have several network interfaces, so it will usually be 0. If you have several network cards used for different purposes, one option would be to configure the cards based on socket position, as in:

```
case "$ADDRESS" in
*,0,*,*)
    # definitions for network card in socket 0
    ;;
*,1,*,*)
    # definitions for network card in socket 1
    ;;
esac
```

Alternatively, they could be configured using their hardware addresses, as in:

```
case "$ADDRESS" in
*,*,*,00:80:C8:76:00:B1)
    # definitions for a D-Link card
    ;;
*,*,*,08:00:5A:44:80:01)
    # definitions for an IBM card
esac
```

Network device parameters

The following parameters can be defined in `network.opts`:

`IF_PORT`

Specifies the ethernet transceiver type, for certain 16-bit cards that do not autodetect. See ``**man** `ifport`'' and ``**man** `mii-tool`'' for more information.

`BOOTP`

A boolean (y/n) value: indicates if the host's IP address and routing information should be obtained using the BOOTP protocol, with `bootpc` or `pump`.

`DHCP`

A boolean (y/n) value: indicates if the host's IP address and routing information should be obtained from a DHCP server. The network script first looks for `dhcpcd`, then `dhclient`, then `pump`.

DHCP_HOSTNAME

Specifies a hostname to be passed to `dhcpcd` or `pump`, for inclusion in DHCP messages.

IPADDR

The IP address for this interface.

NETMASK, BROADCAST, NETWORK

Basic network parameters: see the networking HOWTO for more information.

GATEWAY

The IP address of a gateway for this host's subnet. Packets with destinations outside this subnet will be routed to this gateway.

DOMAIN

The local network domain name for this host, to be used in creating `/etc/resolv.conf`.

SEARCH

A search list for host name lookup, to be added to `/etc/resolv.conf`. `DOMAIN` and `SEARCH` are mutually exclusive: see ``**man** `resolver`'' for more information.

DNS_1, DNS_2, DNS_3

Host names or IP addresses for nameservers for this interface, to be added to `/etc/resolv.conf`

MOUNTS

A space-separated list of NFS mount points to be mounted for this interface.

IPX_FRAME, IPX_NETNUM

For IPX networks: the frame type and network number, passed to the `ipx_interface` command.

NO_CHECK, NO_FUSER

Boolean (y/n) settings for card eject policy. If `NO_CHECK` is set, then ``cardctl eject'' will shut down a device even if there are open connections. If `NO_FUSER` is set, then the script will not check for busy NFS mounts or kill processes using those mounts.

For example:

```
case "$ADDRESS" in
*,*,*,*)
    IF_PORT="10base2"
```

```
    BOOTP="n"
    IPADDR="10.0.0.1"
    NETMASK="255.255.255.0"
    NETWORK="10.0.0.0"
    BROADCAST="10.0.0.255"
    GATEWAY="10.0.0.1"
    DOMAIN="domain.org"
    DNS_1="dns1.domain.org"
    ;;
esac
```

To automatically mount and unmount NFS filesystems, first add all these filesystems to `/etc/fstab`, but include `noauto` in the mount options. In `network.opts`, list the filesystem mount points in the `MOUNTS` variable. It is especially important to use either `cardctl` or `cardinfo` to shut down a network card when NFS mounts are active. It is not possible to cleanly unmount NFS filesystems if a network card is simply ejected without warning.

In addition to the usual network configuration parameters, the `network.opts` script can specify extra actions to be taken after an interface is configured, or before an interface is shut down. If `network.opts` defines a shell function called `start_fn`, it will be invoked by the network script after the interface is configured, and the interface name will be passed to the function as its first (and only) argument. Similarly, if it is defined, `stop_fn` will be invoked before shutting down an interface.

The transceiver type for some (mostly old) cards must be manually be selected using the `IF_PORT` setting. This can either be a numeric value, or a keyword identifying the transceiver type. All the network drivers default to either autodetect the interface if possible, or 10baseT otherwise. The `ifport` command can be used to check or set the current transceiver type. For example:

```
# ifport eth0 10base2
#
# ifport eth0
eth0    2 (10base2)
```

Most modern 10/100baseT cards use a ``media independent interface'' (MII) transceiver that automatically selects line speed and duplex setting. The `mii-tool` command can be used to monitor and control the behavior of the MII interface.

Comments about specific cards

- With IBM CCAE and Socket EA cards, the transceiver type (10base2, 10baseT, AUI) needs to be set when the network device is configured. Make sure that the transceiver type reported in the system log matches your connection.
- The Farallon EtherWave is actually based on the 3Com 3c589, with a special transceiver. Though the EtherWave uses 10baseT-style connections, its transceiver requires that the 3c589 be configured in 10base2 mode.
- If you have trouble with an IBM CCAE, NE4100, Thomas Conrad, or Kingston adapter, try increasing the memory access time with the `mem_speed=#` option to the `pcnet_cs` module. An example of how to do this is given in the standard `config.opts` file. Try speeds of up to 1000 (in nanoseconds).
- For the New Media Ethernet adapter, on some systems, it may be necessary to increase the IO port access time with the `io_speed=#` option when the `pcmcia_core` module is loaded. Edit `CORE_OPTS` in the startup script to set this option.

- The multicast support in the New Media Ethernet driver is incomplete. The latest driver will function with multicast kernels, but will ignore multicast packets. Promiscuous mode should work properly.
- The driver used by the IBM and 3Com token ring adapters seems to behave very badly if the cards are not connected to a ring when they get initialized. Always connect these cards to the net before they are powered up. If `ifconfig` reports the hardware address as all 0's, this is likely to be due to a memory window configuration problem.
- Some Linksys, D-Link, and IC-Card 10baseT/10base2 cards have a unique way of selecting the transceiver type that isn't handled by the Linux drivers. One workaround is to boot DOS and use the vendor-supplied utility to select the transceiver, then warm boot Linux. Alternatively, a Linux utility to perform this function is available at http://pcmcia-cs.sourceforge.net/ftp/extras/dlport.c.
- 16-bit PCMCIA cards have a maximum performance of 1.5-2 MB/sec. That means that any 16-bit 100baseT card (i.e., any card that uses the `pcnet_cs`, `3c574_cs`, `smc91c92_cs`, or `xirc2ps_cs` driver) will never achieve full 100baseT throughput. Only CardBus network adapters can fully exploit 100baseT data rates.
- For WaveLAN wireless network adapters, Jean Tourrilhes (`jt@hpl.hp.com`) has put together a wireless HOWTO at http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/.

Diagnosing problems with network adapters

- In 3.1.15 and later PCMCIA releases, the `test_network` script in the `debug-tools` subdirectory of the PCMCIA source tree will spot some common problems.
- Is your card recognized as an ethernet card? Check the system log and make sure that `cardmgr` identifies the card correctly and starts up one of the network drivers. If it doesn't, your card might still be usable if it is compatible with a supported card. This will be most easily done if the card claims to be ``NE2000 compatible''.
- Is the card configured properly? If you are using a supported card, and it was recognized by `cardmgr`, but still doesn't work, there might be an interrupt or port conflict with another device. Find out what resources the card is using (from the system log), and try excluding these in `/etc/pcmcia/config.opts` to force the card to use something different.
- If your card seems to be configured properly, but sometimes locks up, particularly under high load, you may need to try changing your socket driver timing parameters. See the Startup options section for more information.
- If you get ``Network is unreachable'' messages when you try to access the network, then the routing information specified in `/etc/pcmcia/network.opts` is incorrect. This exact message is an absolutely foolproof indication of a routing error. On the other hand, mis-configured cards will usually fail silently.
- If you are trying to use DHCP to configure your network interface, try testing things with a static IP address instead, to rule out a DHCP configuration problem.
- To diagnose problems in `/etc/pcmcia/network.opts`, start by trying to ping other systems on the same subnet using their IP addresses. Then try to ping your gateway, and then machines on other subnets. Ping machines by name only after trying these simpler tests.
- Make sure your problem is really a PCMCIA one. It may help to see see if the card works under DOS with the vendor's drivers. Double check your modifications to the `/etc/pcmcia/network.opts` script. Make sure your drop cable, ``T'' jack, terminator, etc are working.
- Use real network cables. Don't even think about using that old phone cord you found in your basement. And this means Category 5 cable for 100baseT. It really matters.

4.4 PCMCIA serial and modem devices

Linux serial devices are accessed via the `/dev/ttyS*` and `/dev/cua*` special device files. In pre-2.2 kernels, the `ttyS*` devices were for incoming connections, such as directly connected terminals, and the `cua*` devices were for outgoing connections, such as modems. Use of `cua*` devices is deprecated in current kernels, and `ttyS*` can be used for all applications. The configuration of a serial device can be examined and modified with the `setserial` command.

When a serial or modem card is detected, it will be assigned to the first available serial device slot. This will usually be `/dev/ttyS1` (`cua1`) or `/dev/ttyS2` (`cua2`), depending on the number of built-in serial ports. The `ttyS*` device is the one reported in `stab`. The default serial device option script, `/etc/pcmcia/serial.opts`, will link the device file to `/dev/modem` as a convenience. For pre-2.2 kernels, the link is made to the `cua*` device.

Do not try to use `/etc/rc.d/rc.serial` to configure a PCMCIA modem. This script should only be used to configure non-removable devices. Modify `/etc/pcmcia/serial.opts` if you want to do anything special to set up your modem. Also, do not try to change the IO port and interrupt settings of a serial device using `setserial`. This would tell the serial driver to look for the device in a different place, but would not change how the card's hardware is actually configured. The serial configuration script allows you to specify other `setserial` options, as well as whether a line should be added to `/etc/inittab` for this port.

The device address passed to `serial.opts` has three comma-separated fields: the first is the scheme, the second is the socket number, and the third is the device instance. The device instance may take several values for cards that support multiple serial ports, but for single-port cards, it will always be 0. If you commonly use more than one modem, you may want to specify different settings based on socket position, as in:

```
case "$ADDRESS" in
*,0,*)
    # Options for modem in socket 0
    LINK=/dev/modem0
    ;;
*,1,*)
    # Options for modem in socket 1
    LINK=/dev/modem1
    ;;
esac
```

If a PCMCIA modem is already configured when Linux boots, it may be incorrectly identified as an ordinary built-in serial port. This is harmless, however, when the PCMCIA drivers take control of the modem, it will be assigned a different device slot. It is best to either parse `stab` or use `/dev/modem`, rather than expecting a PCMCIA modem to always have the same device assignment.

If you configure your kernel to load the basic Linux serial port driver as a module, you must edit `/etc/pcmcia/config` to indicate that this module must be loaded. Edit the serial device entry to read:

```
device "serial_cs"
  class "serial" module "misc/serial", "serial_cs"
```

Serial device parameters

The following parameters can be defined in `serial.opts`:

LINK

Specifies a path for a symbolic link to be created to the ``callout'' device (e.g., `/dev/cua*` for pre-2.2, or `/dev/ttyS*` for 2.2 kernels).

SERIAL_OPTS

Specifies options to be passed to the `setserial` command.

INITTAB

If specified, this will be used to construct an `inittab` entry for the device.

NO_CHECK, NO_FUSER

Boolean (y/n) settings for card eject policy. If `NO_CHECK` is true, then ``cardctl eject'' will shut down a device even if it is busy. If `NO_FUSER` is true, then the script will not try to kill processes using an ejected device.

For example:

```
case "$ADDRESS" in
*,*,*)
    LINK="/dev/modem"
    SERIAL_OPTS=""
    INITTAB="/sbin/getty"
```

Comments about specific cards

- The Uniden Data 2000 Wireless CDPD card has some special dialing strings for initiating SLIP and PPP mode. For SLIP, use ``ATDT2''; for PPP, "ATDT0".
- Socket IO revision H serial port cards have a faster-than-normal clock rate for the UART. The card's actual baud rate is four times faster than the serial driver thinks it is. To work around the problem, specify `SERIAL_OPTS="baud_base 460800"` in `/etc/pcmcia/serial.opts`.

Diagnosing problems with serial devices

- In 3.1.15 and later PCMCIA releases, the `test_modem` script in the `debug-tools` subdirectory of the PCMCIA source tree will spot some common problems.
- Is your card recognized as a modem? Check the system log and make sure that `cardmgr` identifies the card correctly and starts up the `serial_cs` driver. If it doesn't, you may need to add a new entry to your `/etc/pcmcia/config` file so that it will be identified properly. See the Configuring unrecognized cards section for details.
- Is the modem configured successfully by serial_cs? Again, check the system log and look for messages from the serial_cs driver. If you see ``register_serial() failed'', you may have an I/O port conflict with another device. Another tip-off of a conflict is if the device is reported to be an 8250; most modern modems should be identified as 16550A UART's. If you think you're seeing a port conflict, edit `/etc/pcmcia/config.opts` and exclude the port range that was allocated for the

modem.
- Is there an interrupt conflict? If the system log looks good, but the modem just doesn't seem to work, try using `setserial` to change the irq to 0, and see if the modem works. This causes the serial driver to use a slower polled mode instead of using interrupts. If this seems to fix the problem, it is likely that some other device in your system is using the interrupt selected by serial_cs. You should add a line to `/etc/pcmcia/config.opts` to exclude this interrupt.
- If the modem seems to work only very, very slowly, this is an almost certain indicator of an interrupt conflict.
- Make sure your problem is really a PCMCIA one. It may help to see if the card works under DOS with the vendor's drivers. Also, don't test the card with something complex like SLIP or PPP until you are sure you can make simple connections. If simple things work but SLIP does not, your problem is most likely with SLIP, not with PCMCIA.
- If you get kernel messages indicating that the serial_cs module cannot be loaded, it means that your kernel does not have serial device support. If you have compiled the serial driver as a module, you must modify `/etc/pcmcia/config` to indicate that the `serial` module should be loaded before `serial_cs`.

4.5 PCMCIA parallel port devices

The Linux parallel port driver is layered so that several high-level device types can share use of the same low level port driver. Printer devices are accessed via the `/dev/lp*` special device files. The configuration of a printer device can be examined and modified with the `tunelp` command.

The `parport_cs` module depends on the `parport` and `parport_pc` drivers, which may be either compiled into the kernel or compiled as modules. The layered driver structure means that any top-level parallel drivers (such as the plip driver, the printer driver, etc) must be compiled as modules. These drivers only recognize parallel port devices at module startup time, so they need to be loaded after any PC Card parallel devices are configured.

The device address passed to `parport.opts` has three comma-separated fields: the first is the scheme, the second is the socket number, and the third is the device instance. The device instance may take several values for cards that support multiple parallel ports, but for single-port cards, it will always be 0. If you commonly use more than one such card, you may want to specify different settings based on socket position, as in:

```
case "$ADDRESS" in
*,0,*)
    # Options for card in socket 0
    LINK=/dev/printer0
    ;;
*,1,*)
    # Options for card in socket 1
    LINK=/dev/printer1
    ;;
esac
```

Parallel device parameters

The following parameters can be defined in `parport.opts`:

LINK

Specifies a path for a symbolic link to be created to the printer port.

LP_OPTS

Specifies options to be passed to the `tunelp` command.

NO_CHECK, NO_FUSER

Boolean (y/n) settings for card eject policy. If `NO_CHECK` is true, then ``cardctl eject'' will shut down a device even if it is busy. If `NO_FUSER` is true, then the script will not try to kill processes using an ejected device.

For example:

```
case "$ADDRESS" in
*,*,*,*)
    LINK="/dev/printer"
    LP_OPTS=""
```

Diagnosing problems with parallel port devices

- Is there an interrupt conflict? If the system log looks good, but the port just doesn't seem to work, try using `tunelp` to change the irq to 0, and see if things improve. This switches the driver to polling mode. If this seems to fix the problem, it is likely that some other device in your system is using the interrupt selected by parport_cs. You should add a line to `/etc/pcmcia/config.opts` to exclude this interrupt.
- If you get kernel messages indicating that the `parport_cs` module cannot be loaded, it means that your kernel does not have parallel device support. If you have compiled the parallel driver as a module, you may need to modify `/etc/pcmcia/config` to indicate that the `parport` and `parport_pc` modules should be loaded before `parport_cs`.

4.6 PCMCIA SCSI adapters

All the currently supported PCMCIA SCSI cards are work-alikes of one of the following ISA bus cards: the Qlogic, the Adaptec AHA-152X, or the Future Domain TMC-16x0. The PCMCIA drivers are built by linking some PCMCIA-specific code (in `qlogic_cs.c`, `aha152x_cs.c`, or `fdomain_cs.c`) with the normal Linux SCSI driver, pulled from the Linux kernel source tree. The Adaptec APA1480 CardBus driver is based on the kernel aic7xxx PCI driver. Due to limitations in the Linux SCSI driver model, only one removable card per driver is supported.

When a new SCSI host adapter is detected, the SCSI drivers will probe for devices. Check the system log to make sure your devices are detected properly. New SCSI devices will be assigned to the first available SCSI device files. The first SCSI disk will be `/dev/sda`, the first SCSI tape will be `/dev/st0`, and the first CD-ROM will be `/dev/scd0`.

A list of SCSI devices connected to this host adapter will be shown in `stab`, and the SCSI configuration script, `/etc/pcmcia/scsi`, will be called once for each attached device, to either configure or shut down that device. The default script does not take any actions to configure SCSI devices, but will properly unmount filesystems on SCSI devices when a card is removed.

The device addresses passed to `scsi.opts` are complicated, because of the variety of things that can be attached to a SCSI adapter. Addresses consist of either six or seven comma-separated fields: the current scheme, the device type, the socket number, the SCSI channel, ID, and logical unit number, and optionally, the partition number. The device type will be ``sd'' for disks, ``st'' for tapes, ``sr'' for CD-ROM devices, and ``sg'' for generic SCSI devices. For most setups, the SCSI channel and logical unit number will be 0. For disk devices with several partitions, `scsi.opts` will first be called for the whole device, with a five-field address. The script should set the `PARTS` variable to a list of partitions. Then, `scsi.opts` will be called for each partition, with the longer six-field addresses.

If your kernel does not have a top-level driver (disk, tape, etc) for a particular SCSI device, then the device will not be configured by the PCMCIA drivers. As a side effect, the device's name in `stab` will be something like ``sd#nnnn'' where ``nnnn'' is a four-digit hex number. This happens when `cardmgr` is unable to translate a SCSI device ID into a corresponding Linux device name.

It is possible to modularize the top-level SCSI drivers so that they are loaded on demand. To do so, you need to edit `/etc/pcmcia/config` to tell `cardmgr` which extra modules need to be loaded when your adapter is configured. For example:

```
device "aha152x_cs"
  class "scsi" module "scsi/scsi_mod", "scsi/sd_mod", "aha152x_cs"
```

would say to load the core SCSI module and the top-level disk driver module before loading the regular PCMCIA driver module.

Always turn on SCSI devices before powering up your laptop, or before inserting the adapter card, so that the SCSI bus is properly terminated when the adapter is configured. Also be very careful about ejecting a SCSI adapter. Be sure that all associated SCSI devices are unmounted and closed before ejecting the card. The best way to ensure this is to use either `cardctl` or `cardinfo` to request card removal before physically ejecting the card. For now, all SCSI devices should be powered up before plugging in a SCSI adapter, and should stay connected until after you unplug the adapter and/or power down your laptop.

There is a potential complication when using these cards that does not arise with ordinary ISA bus adapters. The SCSI bus carries a ``termination power'' signal that is necessary for proper operation of ordinary passive SCSI terminators. PCMCIA SCSI adapters do not supply termination power, so if it is required, an external device must supply it. Some external SCSI devices may be configured to supply termination power. Others, such as the Zip Drive and the Syquest EZ-Drive, use active terminators that do not depend on it. In some cases, it may be necessary to use a special terminator block such as the APS SCSI Sentry 2, which has an external power supply. When configuring your SCSI device chain, be aware of whether or not any of your devices require or can provide termination power.

SCSI device parameters

The following parameters can be defined in `scsi.opts`:

LINK

Specifies a path for a symbolic link to be created to this device.

DO_FSTAB

A boolean (y/n) setting: specifies if an entry should be added to `/etc/fstab` for this device.

DO_FSCK

A boolean (y/n) setting: specifies if the filesystem should be checked before being mounted, with ``fsck -Ta''.

DO_MOUNT

A boolean (y/n) setting: specifies if this device should be automatically mounted at card insertion time.

FSTYPE, OPTS, MOUNTPT

The filesystem type, mount options, and mount point to be used for the fstab entry and/or mounting the device.

NO_CHECK, NO_FUSER

Boolean (y/n) settings for card eject policy. If `NO_CHECK` is true, then ``cardctl eject'' will shut down a device even if it is busy. If `NO_FUSER` is true, then the script will not try to kill processes using an ejected device.

For example, here is a script for configuring a disk device at SCSI ID 3, with two partitions, and a CD-ROM at SCSI ID 6:

```
case "$ADDRESS" in
*,sd,*,0,3,0)
    # This device has two partitions...
    PARTS="1 2"
    ;;
*,sd,*,0,3,0,1)
    # Options for partition 1:
    #  update /etc/fstab, and mount an ext2 fs on /usr1
    DO_FSTAB="y" ; DO_FSCK="y" ; DO_MOUNT="y"
    FSTYPE="ext2"
    OPTS=""
    MOUNTPT="/usr1"
    ;;
*,sd,*,0,3,0,2)
    # Options for partition 2:
    #  update /etc/fstab, and mount an MS-DOS fs on /usr2
    DO_FSTAB="y" ; DO_FSCK="y" ; DO_MOUNT="y"
    FSTYPE="msdos"
    OPTS=""
    MOUNTPT="/usr2"
    ;;
*,sr,*,0,6,0)
    # Options for CD-ROM at SCSI ID 6
    PARTS=""
    DO_FSTAB="y" ; DO_FSCK="n" ; DO_MOUNT="y"
    FSTYPE="iso9660"
    OPTS="ro"
    MOUNTPT="/cdrom"
```

```
    ;;
esac
```

Comments about specific cards

- The Adaptec APA-1480 CardBus card needs a large IO port window (256 contiguous ports aligned on a 256-port boundary). It may be necessary to expand the IO port regions in `/etc/pcmcia/config.opts` to guarantee that such a large window can be found.
- The Adaptec APA-460 SlimSCSI adapter is not supported. This card was originally sold under the Trantor name, and when Adaptec merged with Trantor, they continued to sell the Trantor card with an Adaptec label. The APA-460 is not compatible with any existing Linux driver.
- I have had one report of a bad interaction between the New Media Bus Toaster and a UMAX Astra 1200s scanner. Due to the complexity of the SCSI protocol, when diagnosing problems with SCSI devices, it is worth considering that incompatible combinations like this may exist and may not be documented.

Diagnosing problems with SCSI adapters

- With the `aha152x_cs` driver (used by Adaptec, New Media, and a few others), it seems that SCSI disconnect/reconnect support is a frequent source of trouble with tape drives. To disable this ``feature,'' add the following to `/etc/pcmcia/config.opts`:

  ```
  module "aha152x_cs" opts "reconnect=0"
  ```

- Also with the `aha152x_cs` driver, certain devices seem to require a longer startup delay, controlled via the `reset_delay` module parameter. The Yamaha 4416S CDR drive is one such device. The result is the device is identified successfully, then hangs the system. In such cases, try:

  ```
  module "aha152x_cs" opts "reset_delay=500"
  ```

- Another potential source of SCSI device probe problems is probing of multiple LUN's. If you see successful detection of a device, followed by SCSI bus timeouts when LUN 1 for that device is probed, then disable the kernel's `CONFIG_SCSI_MULTI_LUN` option.
- If you have compiled SCSI support as modules (`CONFIG_SCSI` is ``m''), you may need to modify `/etc/pcmcia/config` to load the SCSI modules before the appropriate `*_cs` driver is loaded.
- If you get ``aborting command due to timeout'' messages when the SCSI bus is probed, you almost certainly have an interrupt conflict.
- If the host driver reports ``no SCSI devices found'', verify that your kernel was compiled with the appropriate top-level SCSI drivers for your devices (i.e., disk, tape, CD-ROM, and/or generic). If a top-level driver is missing, devices of that type will be ignored.

4.7 PCMCIA memory cards

The `memory_cs` driver handles all types of memory cards, as well as providing direct access to the PCMCIA memory address space for cards that have other functions. When loaded, it creates a combination of character and block devices. See the **man** page for the module for a complete description of the device naming scheme. Block devices are used for disk-like access (creating and mounting filesystems, etc). The character devices are for "raw" unbuffered reads and writes at arbitrary locations.

The device address passed to `memory.opts` consists of two fields: the scheme, and the socket number. The options are applied to the first common memory partition on the corresponding memory card.

Some flash memory cards, and most simple static RAM cards, lack a ``Card Information Structure'' (CIS), which is the system PCMCIA cards use to identify themselves. Normally, `cardmgr` will assume that any card that lacks a CIS is a simple memory card, and load the `memory_cs` driver. Thus, a common side effect of a general card identification problem is that other types of cards may be misdetected as memory cards.

There is another issue to consider when handling memory cards that do not have CIS information. At startup time, the PCMCIA package tries to use the first detected card to determine what memory regions are usable for PCMCIA. The memory scan can be fooled if that card is a simple memory card. If you plan to use memory cards often, it is best to limit the memory windows in `/etc/pcmcia/config.opts` to known-good regions.

The `memory_cs` driver uses a heuristic to guess the capacity of these cards. The heuristic does not work for write protected cards, and may make mistakes in some other cases as well. If a card is misdetected, its size should then be explicitly specified when using commands such as `dd` or `mkfs`. The `memory_cs` module also has a parameter for overriding the size detection. See the **man** page.

Memory device parameters

The following parameters can be specified in `memory.opts`:

DO_FSTAB

A boolean (y/n) setting: specifies if an entry should be added to `/etc/fstab` for this device.

DO_FSCK

A boolean (y/n) setting: specifies if the filesystem should be checked before being mounted, with ``fsck -Ta''.

DO_MOUNT

A boolean (y/n) setting: specifies if this device should be automatically mounted at card insertion time.

FSTYPE, OPTS, MOUNTPT

The filesystem type, mount options, and mount point to be used for the fstab entry and/or mounting the device.

NO_CHECK, NO_FUSER

Boolean (y/n) settings for card eject policy. If `NO_CHECK` is true, then ``cardctl eject'' will shut down a device even if it is busy. If `NO_FUSER` is true, then the script will not try to kill processes using an ejected device.

Here is an example of a script that will automatically mount memory cards based on which socket they are inserted into:

```
case "$ADDRESS" in
*,0,0)
    # Mount filesystem, but don't update /etc/fstab
    DO_FSTAB="n" ; DO_FSCK="y" ; DO_MOUNT="y"
    FSTYPE="ext2" ; OPTS=""
    MOUNTPT="/mem0"
    ;;
*,1,0)
    # Mount filesystem, but don't update /etc/fstab
    DO_FSTAB="n" ; DO_FSCK="y" ; DO_MOUNT="y"
    FSTYPE="ext2" ; OPTS=""
    MOUNTPT="/mem1"
    ;;
esac
```

Using linear flash memory cards

The following information applies only to so-called ``linear flash'' memory cards. Many flash cards, including all SmartMedia and CompactFlash cards, actually include circuitry to emulate an IDE disk device. Those cards are thus handled as IDE devices, not memory cards.

There are two major formats for flash memory cards: the FTL or ``flash translation layer'' style, and the Microsoft Flash File System. The FTL format is generally more flexible because it allows any ordinary high-level filesystem (ext2, ms-dos, etc) to be used on a flash card as if it were an ordinary disk device. The FFS is a completely different filesystem type. Linux cannot currently handle cards formated with FFS.

To use a flash memory card as an ordinary disk-like block device, first create an FTL partition on the device with the `ftl_format` command. This layer hides the device-specific details of flash memory programming and make the card look like a simple block device. For example:

```
ftl_format -i /dev/mem0c0c
```

Note that this command accesses the card through the ``raw'' memory card interface. Once formatted, the card can be accessed as an ordinary block device via the `ftl_cs` driver. For example:

```
mke2fs /dev/ftl0c0
mount -t ext2 /dev/ftl0c0 /mnt
```

Device naming for FTL devices is tricky. Minor device numbers have three parts: the card number, the region number on that card, and optionally, the partition within that region. A region can either be treated as a single block device with no partition table (like a floppy), or it can be partitioned like a hard disk device. The ``ftl0c0'' device is card 0, common memory region 0, the entire region. The ``ftl0c0p1'' through ``ftl0c0p4'' devices are primary partitions 1 through 4 if the region has been partitioned.

Configuration options for FTL partitions can be given in `ftl.opts`, which is similar in structure to `memory.opts`. The device address passed to `ftl.opts` consists of three or four fields: the scheme, the socket number, the region number, and optionally, the partition number. Most flash cards have just one flash memory region, so the region number will generally always be zero.

Intel Series 100 flash cards use the first 128K flash block to store the cards' configuration information. To prevent accidental erasure of this information, `ftl_format` will automatically detect this and skip the first block when creating an FTL partition.

4.8 PCMCIA ATA/IDE card drives

ATA/IDE drive support is based on the regular kernel IDE driver. This includes SmartMedia and CompactFlash devices: these flash memory cards are set up so that they emulate an IDE interface. The PCMCIA-specific part of the driver is `ide_cs`. Be sure to use `cardctl` or `cardinfo` to shut down an ATA/IDE card before ejecting it, as the driver has not been made ``hot-swap-proof''.

The device addresses passed to **`ide.opts`** consist of either three or four fields: the current scheme, the socket number, the drive's serial number, and an optional partition number. The `ide_info` command can be used to obtain an IDE device's serial number. As with SCSI devices, **`ide.opts`** is first called for the entire device. If **`ide.opts`** returns a list of partitions in the `PARTS` variable, the script will then be called for each partition.

ATA/IDE fixed-disk device parameters

The following parameters can be specified in **`ide.opts`**:

`DO_FSTAB`

A boolean (y/n) setting: specifies if an entry should be added to `/etc/fstab` for this device.

`DO_FSCK`

A boolean (y/n) setting: specifies if the filesystem should be checked before being mounted, with ``fsck -Ta''.

`DO_MOUNT`

A boolean (y/n) setting: specifies if this device should be automatically mounted at card insertion time.

`FSTYPE`, `OPTS`, `MOUNTPT`

The filesystem type, mount options, and mount point to be used for the fstab entry and/or mounting the device.

`NO_CHECK`, `NO_FUSER`

Boolean (y/n) settings for card eject policy. If `NO_CHECK` is true, then ``cardctl eject'' will shut down a device even if it is busy. If `NO_FUSER` is true, then the script will not try to kill processes using an ejected device.

Here is an example **`ide.opts`** file to mount the first partition of any ATA/IDE card on `/mnt`.

```
case "$ADDRESS" in
```

```
*,*,*,1)
    DO_FSTAB="y" ; DO_FSCK="y" ; DO_MOUNT="y"
    FSTYPE="msdos"
    OPTS=""
    MOUNTPT="/mnt"
    ;;
*,*,*)
    PARTS="1"
    ;;
esac
```

Diagnosing problems with ATA/IDE adapters

- An IO port conflict may cause the IDE driver to misdetect the drive geometry and report ``INVALID GEOMETRY: 0 PHYSICAL HEADS?''. To fix, try excluding the selected IO port range in `/etc/pcmcia/config.opts`.
- Some IDE drives violate the PCMCIA specification by requiring more time to spin up than the maximum allowed card setup time. This may result in ``timed out during reset'' messages at card detect time. Adjust the `unreset_delay` and/or `unreset_limit` parameters for the `pcmcia_core` module to give a drive more time to spin up; see the `pcmcia_core` **man** page for parameter details. For example:

  ```
  CORE_OPTS="unreset_delay=400"
  ```

- To use an ATA/IDE CD-ROM device, your kernel must be compiled with `CONFIG_BLK_DEV_IDECD` enabled. This will normally be the case for standard kernels, however it is something to be aware of if you compile a custom kernel.
- A common error when using IDE drives is try to mount the wrong device file. Generally, you want to mount a partition of the device, not the entire device (i.e., `/dev/hde1`, not `/dev/hde`).
- The Linux IDE driver may have trouble configuring certain removable-media drives if no media is present at insertion time. The IBM Portable DriveBay has this problem.
- Some kernels will report a pair of ``drive_cmd'' errors at insertion time. These errors can be ignored: they pop up when a removable IDE device does not accept the IDE ``door lock'' command.

4.9 Multifunction cards

A single interrupt can be shared by several drivers, such as the serial driver and an ethernet driver: in fact, the PCMCIA specification requires all card functions to share the same interrupt. Normally, all card functions are available without having to swap drivers. All Linux kernels support this kind of interrupt sharing.

Simultaneous use of two card functions is ``tricky'' and various hardware vendors have implemented interrupt sharing in their own incompatible (and sometimes proprietary) ways. The drivers for some cards (Ositech Jack of Diamonds, 3Com 3c562 and related cards, Linksys cards) properly support simultaneous access, but others (older Megahertz cards in particular) do not. If you have trouble using a card with both functions active, try using each function in isolation. That may require explicitly doing an ```ifconfig down`'' to shut down a network interface and use a modem on the same card.

Next Previous Contents