Linux PCMCIA HOWTO

David Hinds, dhinds@allegro.stanford.edu

v1.44, 1995/07/31 05:50:14

This document describes how to install and use PCMCIA Card Services for Linux, and answers some frequently asked questions. The latest version of this document can always be found at cb-iris.stanford.edu in /pub/pcmcia/doc. An HTML version is at http://hyper.stanford.edu/~dhinds/pcmcia/pcmcia.html

Contents

1 General information and hardware requirements

1.1 Introduction

Card Services for Linux is a complete PCMCIA support package. It includes a set of loadable kernel modules that implement a version of the PCMCIA Card Services applications program interface, a set of client drivers for specific cards, and a card manager daemon that can respond to card insertion and removal events, loading and unloading drivers on demand. It supports "hot swapping" of PCMCIA cards, so cards can be inserted and ejected at any time.

This is beta software. It probably contains bugs, and should be used with caution. I'll do my best to fix problems that are reported to me, but if you don't tell me, I may never know. If you use this code, I hope you will send me your experiences, good or bad!

If you have any suggestions for how this document could be improved, please let me know (`dhinds@allegro.stanford.edu`).

1.2 Copyright notice and disclaimer

1.3 What is the latest version, and where can I get it?

The current release of Card Services is version 2.6.3.

The latest version is always available from `cb-iris.stanford.edu` in the `/pub/pcmcia` directory. There will sometimes be several versions here. In that case, the oldest version should be more stable, and newer versions generally contain more experimental code. It is up to you to decide which version is more appropriate, but the `CHANGES` file will summarize the most important differences.

`cb-iris.stanford.edu` is mirrored at `sunsite.unc.edu` in `/pub/Linux/kernel/pcmcia`. I'll also try to upload major releases to `tsx-11.mit.edu` under `/pub/linux/laptops/pcmcia/drivers` now and then.

1.4 What systems are supported?

This code should run on almost any Linux-capable laptop. All common PCMCIA controllers are supported, including Intel, Cirrus, Vadem, VLSI, Ricoh, and Databook chips. Custom controllers used in IBM and Toshiba laptops are also supported. Several people use the package on desktop systems with PCMCIA card adapters.

The Motorola 6AHC05GA controller used in some Hyundai laptops is not supported. The custom PCMCIA controller in the HP Omnibook 600 is also unsupported.

1.5 What PCMCIA cards are supported?

The current release includes drivers for a variety of ethernet cards, a driver for modem and serial port cards, several SCSI adapter drivers, and memory card drivers that should support most SRAM cards and some flash cards. The `SUPPORTED.CARDS` file included with each release of Card Services lists all cards that are known to work in at least one actual system.

The likelihood that a card not on the supported list will work depends on the type of card. Essentially all modems should work with the supplied driver. Some network cards may work if they are OEM versions of supported cards. Other types of IO cards (hard drives, sound cards, etc) will not work until someone writes the appropriate drivers.

1.6 When will card X be supported?

Unfortunately, they don't pay me to write device drivers, so if you'd like to have a driver for your favorite card, you're probably going to have to do some of the work on your own. The `SUPPORTED.CARDS` file mentions some cards for which driver work is currently in progress. I will try to help where I can.

1.7 Mailing list

I maintain a database and mailing list of Linux PCMCIA users. This is used to announce new releases of the PCMCIA package. If you would like to be included, send me the following:

- Your name and email address
- What kind of laptop are you using?
- What PCMCIA controller is reported by the `probe` command?
- What PCMCIA cards are you using?
- Any special settings you use: compilation options, irq and port settings, `/etc/pcmcia/config` entries, `insmod` options, etc.

You can also register via the WWW: see `http://hyper.stanford.edu/~dhinds/pcmcia/pcmcia.html` for instructions.

2 Compilation, installation, and configuration

2.1 Prerequisites and kernel setup

For the latest version, you will need to have kernel version 1.2.8 or higher. There are no kernel patches specifically for PCMCIA support. You'll also need to have a relatively recent set of module utilities. If your man page for `insmod` describes the `[symbol=value ...]` syntax, your utilities are current enough.

You need to have a complete linux source tree for your kernel, not just an up-to-date kernel image, when you compile the PCMCIA package. The PCMCIA modules contain some references to kernel source files.

Current kernel sources and patches are available from `sunsite.unc.edu` in `/pub/Linux/kernel/v1.2`, or from `tsx-11.mit.edu` in `/pub/linux/sources/system/v1.2`. Current module utilities can be found in the same places, in the file `modules-1.2.8.tgz`.

When configuring your kernel, if you plan on using a PCMCIA ethernet card, you should turn on networking support but turn off the normal Linux network card drivers, including the "pocket and portable adapters". The PCMCIA network card drivers are all implemented as loadable modules. All of the PCMCIA net drivers except the 3Com 3c589 driver depend on the `8390.o` driver module which is built as part of the Linux kernel.

If you want to use SLIP, PPP, or PLIP, you do need to either configure your kernel with these enabled, or use the loadable module versions of these drivers. There is an unfortunate deficiency in the kernel config process, in that it is not possible to set configuration options (like SLIP compression) for a loadable module, so it is probably better to just link SLIP into the kernel if you need it.

If you will be using a PCMCIA SCSI adapter, you should enable CONFIG_SCSI when configuring your kernel. Also, enable any top level drivers (SCSI disk, tape, cdrom, generic) that you expect to use. All low-level drivers for particular host adapters should be disabled.

For recent kernels, you must explicitly do "`make modules`" followed by "`make modules_install`" in `/usr/src/linux` to build the loadable driver modules. They will be installed under `/lib/modules`.

Building the PCMCIA modules requires access to the kernel version file `versions.h`. This file is normally created at the start of a kernel build, and deleted at the end. It will be recreated in the course of doing "`make modules`", or it can be explicitly created by doing "`make include/linux/version.h`" at the top of your Linux source tree.

2.2 Installation

Starting with release 2.4.8, this package includes an X-based card status utility called `cardinfo`. This utility is based on a public domain user interface toolkit called the Forms Library, which you will need to install before building `cardinfo`. A binary distribution is on `cb-iris.stanford.edu` in `/pub/pcmcia/extras/bxform-0.61.tgz`. There is a small bug in the Makefile: the line that starts with "`ln -s`" should have "`; fi`" added to the end.

Unpack the `pcmcia-cs-2.6.3.tgz` package in a convenient location, like `/usr/src`.

Make sure the definitions in `make.options` are consistent with your site setup. Running "`make prereq`" will check your system configuration to verify that it satisfies all prerequisites for installing PCMCIA support.

Running "`make all`" followed by "`make install`" will build and then install the kernel modules and utility programs. Kernel modules are installed under `/lib/modules/<version>/pcmcia`. The `cardmgr` and `cardctl` programs are installed in `/sbin`. If `cardinfo` is built, it is installed in `/usr/bin/X11`.

Configuration files will be installed in the `/etc/pcmcia` directory. If you are installing over an older version, the new config files will be installed with a "`.N`" suffix – you should replace or update your existing files by hand.

If you don't know what kind of PCMCIA controller chip you have, you can use the `probe` utility in the `cardmgr/` subdirectory to determine this. There are two major types: the Databook TCIC-2 type and the Intel i82365SL-compatible type.

A user-level daemon processes card insertion and removal events. This is called `cardmgr`. It is similar in function to Barry Jaspan's `pcmciad` in earlier PCMCIA releases. `Cardmgr` reads a configuration file describing known PCMCIA cards from `/etc/pcmcia/config`. This file also specifies what resources can be allocated for use by PCMCIA devices, and may need to be customized for your system. See the `pcmcia` man page for more information about this file.

The script `rc.pcmcia`, installed in `/etc/rc.d`, controls starting up and shutting down the PCMCIA system. "`make install`" will use the `probe` command to determine your controller type and modify `rc.pcmcia` appropriately. You should add a line to your system startup file `/etc/rc.d/rc.M` to invoke this:

```
/etc/rc.d/rc.pcmcia start
```

In a few cases, the `probe` command will be unable to determine your controller type automatically. The Tadpole P1000 and some other PCI-based laptops have a special Cirrus PCI-to-PCMCIA bridge chip that can't be detected by `probe`. If you have one of these systems, you'll need to install `rc.pcmcia` by hand.

2.3 Site-specific configuration options

Card Services should automatically avoid allocating IO ports and interrupts already in use by other standard devices. This should work for any devices that have Linux drivers, like serial and parallel ports, IDE drives, and some sound cards. If a device is unsupported by Linux, you may need to explicitly exclude the resources it uses in `/etc/pcmcia/config`.

- On the AMS SoundPro, exclude irq 10.
- On the BMX 486DX2-66, exclude irq 5, irq 9.
- On the NEC Versa M, exclude irq 9, port 0x2e0-2ff.
- On the NEC Versa P/75, exclude irq 5, irq 9.
- On the NEC Versa S, exclude irq 9, irq 12.
- On the ProStar 9200, Altima Virage, and Acquiline Hurricane DX4-100, exclude irq 5, port 0x330-0x35f. Maybe use memory 0xd8000-0xdffff.
- On the Toshiba T4900 CT, exclude irq 5, port 0x2e0-0x2e8, port 0x330-0x338.
- On the Twinhead 5100, HP 4000, Sharp PC-8700 and PC-8900, exclude irq 9 (sound), irq 12.
- On an MPC 800 Series, exclude irq 5, port 0x300-0x30f for the CD-ROM.

Some PCMCIA controllers have optional features that may or may not be implemented in a particular system. It is generally impossible for a socket driver to detect if these features are implemented. Check the man page for your driver to see what optional features may be enabled.

The low level socket drivers, `tcic` and `i82365`, have numerous bus timing parameters that may need to be adjusted for systems with particularly fast processors. Symptoms of timing problems include card recognition problems, lock-ups under heavy loads, high error rates, or poor device performance. Check the corresponding man pages for more details, but here is a brief summary:

- Cirrus controllers have numerous configurable timing parameters. The most important is the `freq_bypass` flag which changes the multiplier for the PCMCIA bus clock to slow down all operations.
- The Cirrus PD6729 PCI controller has the `fast_pci` flag, which should be set if the PCI bus speed is greater than 25 MHz.

- For Vadem VG-468 controllers and Databook TCIC-2 controllers, the `async_clock` flag changes the relative clocking of PCMCIA bus and host bus cycles. Setting this flag adds extra wait states to some operations.
- The `pcmcia_core` module has the `cis_speed` parameter for changing the memory speed used for accessing a card's Card Information Structure (CIS). On some systems with fast bus clocks, increasing this parameter (i.e., slowing down card accesses) may be beneficial.

All these options should be configured by modifying the top of `/etc/rc.d/rc.pcmcia`. For example:

```
# Should be either i82365 or tcic
PCIC=i82365
# Put socket driver timing parameters here
OPTS="async_clock=1"
```

On some systems using Cirrus controllers, including the NEC Versa M, the BIOS puts the controller in a special suspended state at system startup time. On these systems, the `probe` command will fail to find any known PCMCIA controller. If this happens, edit `/etc/rc.d/rc.pcmcia` by hand as follows:

```
# Should be either i82365 or tcic
PCIC=i82365
# Put socket driver timing parameters here
OPTS="wakeup=1"
```

If you have an ARM Pentium-90 or Midwest Micro Soundbook Plus laptop, use the combination "`freq_bypass=1 cmd_time=8`" to slow down your PCMCIA bus cycles. On a Midwest Micro Soundbook Elite, try "cmd_time=12". These may help on other very fast systems that use the non-PCI Cirrus chip (the PD672x).

2.4 What about installation on Caldera?

Caldera has a System V-ish arrangement for system startup files. The PCMCIA installation scripts will automatically detect this and adjust accordingly. The `rc.pcmcia` script will be installed as `/etc/rc.d/init.d/pcmcia`. There is no need to edit any of the Caldera startup scripts to enable PCMCIA: it will happen automatically. Also, the PCMCIA configuration scripts will be installed under `/etc/sysconfig/pcmcia-scripts`, instead of `/etc/pcmcia`.

A separate configuration file, `/etc/sysconfig/pcmcia`, will be created for startup options. If you need to change any module options (like the `PCIC=` or `OPTS=` settings), edit this config file rather than the actual PCMCIA startup script. This file will not be overwritten by subsequent installs.

2.5 Can I install Linux via NFS with a PCMCIA network card?

I've created a set of 1.44MB boot and root disks with PCMCIA support for the Slackware 2.2 distribution. The files are `pcboot14.gz` and `pcroot14.gz` on `cb-iris.stanford.edu` and `sunsite.unc.edu` (see section

1.3). The root disk includes `cardmgr`, the core PCMCIA modules, and all the network drivers. As for how to use these, you should familiarize yourself with the Slackware installation instructions, available from the usual FTP sites. The PCMCIA drivers will be loaded automatically, and installation will be the same as for a non-PCMCIA net card. Note that Slackware root disks do not include any normal user-level network utilities (`ftp`, `telnet`, etc). They only include enough network support to establish an NFS mount.

If you use these disks with an IBM Thinkpad, you may need to specify "floppy=thinkpad" at the lilo prompt when you boot the pcboot disk.

After installation is complete, you'll have a non-PCMCIA setup on your root disk. It is possible to copy things from the boot and root disks to your hard disk to get a working network setup, but it is a little tricky to put everything in the right places by hand. Once you have booted your newly installed Linux system from your hard disk, mount the Slackware boot disk on `/mnt`, and do:

```
cp /mnt/vmlinuz /linuz
rootflags /vmlinuz 1
lilo
```

Then, mount the Slackware root disk on /mnt, and do:

```
cp /mnt/sbin/cardmgr /sbin
(cd /mnt ; tar cf - etc/pcmcia lib/modules) | (cd / ; tar xf -)
```

Edit `/etc/pcmcia/config` and un-comment the "start" and "stop" commands for the net card drivers. Rename `/etc/pcmcia/network.sample` to `/etc/pcmcia/network` and edit to conform to your network setup. You will need to edit `/etc/rc.d/rc.M` by hand to start up the PCMCIA stuff as in `/etc/rc.local` on the Slackware root disk.

Alternatively, if your install server has a current set of source files, you can copy current kernel sources, pcmcia sources, and module utilities to your hard disk while it is NFS mounted. Then, after rebooting, build a new kernel and install the PCMCIA software as normal.

The Slackware boot/root disk combination is configured to work in many systems, but no one configuration can work in all situations. It is difficult to debug PCMCIA setup problems encountered with these disks, because of the very limited set of tools available. When `cardmgr` is running, `/etc/stab` will show what cards are configured. At boot time, messages from the PCMCIA modules are hard to spot before they scroll off the screen, so "`/etc/rc.d/rc.pcmcia restart`" may give some useful info.

This disk set is really only meant to be used to install Slackware; I do not recommend trying to use it to avoid having to compile the full PCMCIA support package. The disk images are updated infrequently and are missing several important PCMCIA components.

2.6 Why don't you distribute PCMCIA binaries?

For me, distributing binaries is a significant hassle. It is complicated because some features can only be selected at compile time, and because the PCMCIA modules are somewhat dependent on having the "right" kernel configuration. So, I would probably need to distribute precompiled modules along with matching kernels. Beyond this, the greatest need for precompiled modules is when installing Linux on a clean system. This typically requires setting up PCMCIA so that it can be used in the installation process for a particular Linux distribution. Each Linux distribution has its own procedures, and it is not feasible for me to provide boot and root disks for even just the common combinations of drivers and distributions.

I think binary distributions are best handled by the people who put together distributions, rather than by me. If you would like your favorite Linux bundler to include PCMCIA in their distributions, I encourage you to ask them for it. Several have talked to me already and I expect that PCMCIA will become a standard part of most Linux distributions in the future.

According to the Linux Distribution HOWTO, the Nascent Technology Linux distribution already includes PCMCIA support.

2.7 Why doesn't my system respond to card insertions?

In most cases, the socket driver (`i82365` or `tcic`) will automatically probe and select an appropriate interrupt to signal card status changes. The automatic interrupt probe doesn't work on some Intel-compatible controllers, including Cirrus chips and the chips used in some IBM ThinkPads. If a device is inactive at probe time, its interrupt may also appear to be available. In these cases, the socket driver may pick an interrupt that is used by another device.

With the `i82365` and `tcic` drivers, the `irq_mask` option can be used to limit the interrupts that will be tested. This mask limits the set of interrupts that can be used by PCMCIA cards as well as for monitoring card status changes. The `cs_irq` option can also be used to explicitly set the interrupt to be used for monitoring card status changes.

If you can't find an interrupt number that works, there is also a polled status mode: both `i82365` and `tcic` will accept a `poll_interval=100` option, to poll once per second.

All these options should be set in the `OPTS=` line in either `/etc/rc.d/rc.pcmcia` or `/etc/sysconfig/pcmcia`, depending on your site setup.

3 Usage and features

3.1 How do I tell if it is working?

The `cardmgr` daemon normally beeps when a card is inserted, and the tone of the beeps indicates the status of the newly inserted card. Two high beeps indicate the card was identified and configured successfully. A high beep followed by a lower beep indicates that the card was identified, but could not be configured for some reason. One low beep indicates that the card could not be identified.

If you are running X, the new `cardinfo` utility produces a slick graphical display showing the current status of all PCMCIA sockets.

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

```
Module:         #pages:  Used by:
ds                 2
i82365             2
pcmcia_core        6     [ds i82365]
```

All the PCMCIA modules and the `cardmgr` daemon send status messages to the system log. This will usually be `/usr/adm/messages`. This file should be the first place to look when tracking down a problem. When submitting a bug report, always include the contents of this file. `Cardmgr` also records some current device information for each socket in `/etc/stab`.

You can also submit bug reports via the WWW: see `http://hyper.stanford.edu/~dhinds/pcmcia/pcmcia.html` for details.

3.2 How do I use my PCMCIA ethernet card?

Linux ethernet-type network interfaces normally have names like `eth0`, `eth1`, 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 don't be surprised when you can't find them.

When a PCMCIA ethernet card is detected, it will be assigned the first free interface name, which will probably be `eth0`. `Cardmgr` will run the `/etc/pcmcia/network` script to configure the interface, which should be customized for your local network setup.

Do not configure your PCMCIA ethernet card in `/etc/rc.d/rc.inet1`, since the card may not be present when this script is executed. Comment out everything except the loopback stuff in `rc.inet1` and instead edit the `/etc/pcmcia/network` script to match your local network setup. This script will be executed only when your ethernet card is actually present.

3.3 How do I use my PCMCIA modem card?

Linux serial devices are accessed via the `/dev/cua*` and `/dev/ttyS*` special device files. The `ttyS*` devices are for incoming connections, such as directly connected terminals. The `cua*` devices are for outgoing connections, such as modems. The configuration of a serial device can be examined and modified with the `setserial` command.

When a PCMCIA serial or modem card is detected, it will be assigned to the first available serial device slot. This will usually be `/dev/cua1` or `/dev/cua2`, depending on the number of built-in serial ports. The default serial device script, `/etc/pcmcia/serial`, will link the appropriate device file to `/dev/modem` as a convenience.

If you are using more than one PCMCIA modem, use `/etc/stab` or `cardinfo` to find out which device corresponds to each modem.

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` if you want to do anything special to set up your modem.

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 `/etc/stab` or use `/dev/modem`, rather than expecting a PCMCIA modem to always have the same device assignment.

3.4 How do I use my PCMCIA SCSI card?

The Qlogic FastSCSI, New Media Bus Toaster, and Adaptec APA-1460 SlimSCSI cards work under Card Services. The PCMCIA driver modules for these cards are built by linking some PCMCIA-specific code (in `qlogic_cs.c` and `toaster_cs.c`) with a normal Linux SCSI driver. The Qlogic PCMCIA driver links with the normal QLogic driver. The Bus Toaster PCMCIA driver, which also supports the Adaptec SlimSCSI, links with the Adaptec 152x driver. It is important to use a 1.2.8 or later kernel if you will be using a SCSI adapter, because there have been important changes to the low-level drivers in recent releases.

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'm not sure how hard it would be to write a driver; I don't think anyone has been able to obtain the technical information from Adaptec.

The Trantor SlimSCSI can be identified by the following:

Trantor / Adaptec APA-460 SlimSCSI FCC ID: IE8T460 Shipped with SCSIworks! driver software

The Adaptec SlimSCSI can be identified by the following:

Adaptec APA-1460 SlimSCSI FCC ID: FGT1460 P/N: 900100 Shipped with EZ-SCSI driver software

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.

When a new SCSI host adapter is detected, the SCSI drivers will probe for devices. Check `/usr/adm/messages` 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 CDROM will be `/dev/scd0`.

Be very careful about ejecting a SCSI adapter. Be sure that all associated SCSI devices are unmounted and closed before 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.

Roger Pao (`rpao@paonet.org`) adds:

Make sure there is a device capable of supplying termination power both to the terminating target device and to the PCMCIA SCSI card. This is vitally important as most PCMCIA SCSI cards do not supply termination power to the SCSI bus. This is usually to save laptop battery power and to isolate the laptop from the SCSI bus.

For a recommendation, the APS SCSI Sentry 2 ($100) is an external Centronics active terminator block (male on one end, female on the other) which uses an external power supply (115/230VAC) to supply termination power (5VDC 2.0A) to its own active terminator and to the PCMCIA SCSI card's terminators (be it active or passive). For more details, APS Technical Support can be reached at 800-334-7550.

3.5 How do I use my PCMCIA memory card?

The default memory card startup script will create block and character devices for accessing a card's first common memory and attribute memory regions. Check the man pages for all the details, but the devices you'll probably be using will be `/dev/mem0c` (character device) or `/dev/mem0b` (block device). The block device is used for disk-like access (creating and mounting filesystems, etc). The character device is for "raw" reads and writes at arbitrary locations.

To use a flash memory card as an ordinary disk-like block device, first create a "flash translation layer" partition on the device with the `ftl_format` command:

```
ftl_format -i /dev/mem0c
```

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/ftl0
mount -t ext2 /dev/ftl0 /mnt
```

3.6 How do I tell `cardmgr` how to identify a new card?

Assuming that your card is supported by an existing driver, all that needs to be done is to add an entry to `/etc/pcmcia/config` to tell `cardmgr` how to identify the card, and which driver(s) need to be linked up to this card. Check the man page for `pcmcia` for more information about the config file format. If you insert an unknown card, `cardmgr` will normally record some identification information in `/usr/adm/messages` that can be used to construct the config entry.

Here is an example of how cardmgr will report an unsupported card in `/usr/adm/messages`.

```
cardmgr[460]: unsupported card in socket 1
cardmgr[460]: version info: "MEGAHERTZ", "XJ2288", "V.34 PCMCIA MODEM"
```

The corresponding entry in `/etc/pcmcia/config` would be:

```
card "Megahertz XJ2288 V.34 Fax Modem"
  version "MEGAHERTZ", "XJ2288", "V.34 PCMCIA MODEM"
  bind "serial_cs"
```

You can use "*" to match strings that don't need to match exactly, like version numbers. When making new config entries, be careful to copy the strings exactly, preserving case and blank spaces. Also be sure that the config entry has the same number of strings as are reported in the log file.

After editing `/etc/pcmcia/config`, you can signal `cardmgr` to reload the file with:

```
kill -HUP `cat /var/run/cardmgr.pid`
```

If you do set up an entry for a new card, please send me a copy so that I can include it in the standard config file.

3.7 How do I control which interrupts and ports are used by a device?

In theory, it should not really matter which interrupt is allocated to which device, as long as two devices are not configured to use the same interrupt. At the top of `/etc/pcmcia/config` you'll find a place for excluding interrupts that are used by non-PCMCIA devices.

All the client card drivers have a parameter called `irq_mask` for specifying which interrupts they may try to allocate. Each bit of irq_mask corresponds to one irq line: bit 0 is irq 0, bit 1 is irq 1, and so on. So, a mask of 0x1200 would correspond to irq 9 and irq 12. To limit a driver to use only one specific interrupt, its irq_mask should have only one bit set. These driver options should be set in your `/etc/pcmcia/config` file. For example:

```
device "serial_cs"
  module "serial_cs" opts "irq_mask=0x1100"
  ...
```

would specify that the serial driver should only use irq 8 or irq 12. Note that Card Services will never allocate an interrupt that is already in use by another device, or an interrupt that is excluded in the config file.

There is no way to directly specify the I/O addresses for a PCMCIA card to use. The `/etc/pcmcia/config` file allows you to specify ranges of ports available for use by all PCMCIA devices.

After modifying `/etc/pcmcia/config`, you can restart `cardmgr` with "`kill -HUP`".

3.8 When is it safe to insert or eject a PCMCIA card?

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.

3.9 How do I unload PCMCIA drivers?

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 PCMCIA device is currently in use, the shutdown will fail.

3.10 How does Card Services deal with suspend/resume?

Card Services can be compiled with support for APM (Advanced Power Management) if you've installed this package on your system. The current release of Stephen Rothwell's APM support package is version 0.5. Unlike the 0.4 release, 0.5 does not require a special patch to work with PCMCIA. The PCMCIA modules will automatically be configured for APM if a compatible version is detected on your system.

Without resorting to APM, you can do "`cardctl suspend`" before suspending your laptop, and "`cardctl resume`" after resuming, to properly shut down and restart your PCMCIA cards. This will not work with a PCMCIA 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.

3.11 How do I turn off a PCMCIA card without ejecting it?

Use either the `cardctl` or `cardinfo` command. "`cardctl suspend #`" will suspend one socket, and turn off its power. The corresponding `resume` command will wake up the card in its previous state.

4 Problems with specific cards

4.1 Why doesn't my modem work?

That's a broad question, but here's a quick troubleshooting guide.

- Is your card recognized as a modem? Check `/usr/adm/messages` 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 section 3.6 for details.
- Is the modem configured successfully by serial_cs? Again, check `/usr/adm/messages` 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 PCMCIA modems should be identified as 16550A UART's. If you think you're seeing a port conflict, edit `/etc/pcmcia/config` and exclude the port range that was allocated for the modem.
- Is there an interrupt conflict? If `/usr/adm/messages` 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` to exclude this interrupt.
- If the modem seems to work only really, really 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 until you are sure you can make simple connections. If simple things work but SLIP does not, your problem is with SLIP, not with PCMCIA.

4.2 Why does my Megahertz modem sometimes fail to work?

Earlier versions of the PCMCIA drivers often failed to properly initialize some Megahertz modems, specifically the 2144 model. This problem should be fixed in current releases.

I've also received one report from someone with a newer Megahertz modem that has a 16550-type UART. He says that he wasn't able to get this modem to work under Linux with `cu` until he configured the modem with:

```
echo 'ATS=QV1X4\&C1\&D2S95=2W1\&K3S36=7S95=255' > /dev/modem
```

This initialization string was supplied by Megahertz tech support.

4.3 Why doesn't my ethernet card work?

Here's another quick troubleshooting guide.

- Is your card recognized as an ethernet card? Check `/usr/adm/messages` 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 `/usr/adm/messages`), and try excluding these in `/etc/pcmcia/config` to force the card to use something different.
- With Socket EA and 3Com 3c589 cards, you need to pick the transceiver type (10base2, 10baseT, AUI) when the driver module is loaded. Make sure that the transceiver type reported in `/usr/adm/messages` 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.
- 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` script. Make sure your drop cable, "T" jack, terminator, etc are working.
- 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 section 2.3 for more information.

4.4 How do I select the transceiver type for my 3c589 card?

It would be nice if the driver could autodetect the difference between a 10baseT and a 10base2 connection, but I don't know how to do that. For now, you need to edit /etc/pcmcia/config and add an `if_ports=#`

option to the 3c589_cs module definition. Check the tc589_cs man page for more details, but to select 10base2 (also known as BNC, or thin net, or coax), change:

```
module "3c589_cs"
```

to:

```
module "3c589_cs" opts "if_port=3"
```

4.5 My network performance stinks. What can I do?

If you have an IBM CCAE, NE4100, Thomas Conrad, or Kingston adapter, increase the memory access time with the mem_speed=# option to the ibmcc_cs module definition. Try speeds of up to 1000 (in nanoseconds).

For other cards, you may need to try changing your socket driver timing parameters. Check the man page for your socket driver (i82365 or tcic) to see what parameters are available.

4.6 How do I add support for an NE2000-compatible ethernet card?

First, see if the card is already recognized by cardmgr. Some cards not listed in SUPPORTED.CARDS are actually OEM versions of cards that are supported. If you find a card like this, let me know so I can add it to the list.

If your card is not recognized, follow the instructions in section 3.6 to create a config entry for your card, but bind the card to the memory card driver, pcmem_cs for now. Restart cardmgr to use the new updated config file.

You will need to know your card's hardware ethernet address. This address is a series of six two-digit hex numbers, often printed on the card itself. If it is not printed on the card, you may be able to use a DOS driver to display the address. In any case, once you know it, run:

```
dd if=/dev/pcmem0a count=20 | od -Ax -t x1
```

and search the output for your address. Record the hex offset of the first byte of the address. Now, edit modules/de650_cs.c and find the hw_info structure. You'll need to create a new entry for your card. The first field is the offset multiplied by two. The next three fields are the first three bytes of the hardware address. The final field is just a descriptive name.

After editing de650_cs.c, compile and install the new module. Edit /etc/pcmcia/config again, and change the card binding from pcmem_cs to de650_cs. Follow the instructions for reloading the config file, and you should be all set. Please send me copies of your new hw_info and config entries.

If you can't find your card's hardware address in the hex dump, as a method of last resort, it is possible to "hardwire" the address when the de650_cs module is initialized. Edit /etc/pcmcia/config and add a hw_addr= option, like so:

```
module "de650_cs" opts "hw_addr=0x00,0x80,0xc8,0x80,0x05,0x76"
```

Substitute your own card's hardware address in the appropriate spot, of course.

4.7 How do I use my PCMCIA floppy interface?

The PCMCIA floppy interface used in the Compaq Aero and a few other laptops is not yet supported by this package. If your laptop can initialize this card before Linux boots, you should be able to use it by telling Card Services to ignore that socket. Note that you will not be able to hot swap this card.

The snag in supporting the Aero floppy is that the Aero seems to use a proprietary PCMCIA controller to support DMA to the floppy. Without knowing exactly how this is done, there isn't any way to implement support under Linux.

To configure Card Services to ignore a socket, use the `ignore=#` parameter when you load the `i82365` or `tcic` driver. See the man pages for more details.

4.8 What's up with support for Xircom cards?

Xircom does not share technical information about its cards without a non-disclosure agreement. This means that it is not really possible to develop freely distributable drivers for Xircom cards without doing legally dubious things like reverse engineering DOS drivers.

There is some indication that Xircom may start supporting Linux directly. Xircom tech support says that future products will include Linux drivers. Their plans for older Xircom products are less clear.

The Xircom CreditCard Ethernet+Modem II card can be used as a modem under Linux, with no special configuration.

5 Debugging tips and programming information

5.1 How can I submit a helpful bug report?

Here are some things that should be included in all bug reports:

- Your system type, and the output of the `probe` command
- What PCMCIA cards you are using
- Your Linux kernel version, and PCMCIA version
- Any changes you've made to the startup files in `/etc/pcmcia`
- Contents of `/usr/adm/messages`, even if you don't see anything that looks interesting.

The `make.options` file includes a few choices for building the kernel modules with various kinds of debugging code turned on. This may or may not be useful, depending on your problem. It is probably better to only turn on the really verbose debugging if I ask you to.

If your problem involves a kernel fault, the register dump from the fault is only useful if you can track down the fault address, EIP. If it is in the main kernel, look up the address in `zSystem.map` to identify the function at fault. If the fault is in a loadable module, it is a bit harder to trace. With the current module tools, "`ksyms`

-m" will report the base address of each loadable module. Pick the module that contains the EIP address, and subtract its base address from EIP to get an offset inside that module. Then, run gdb on that module, and look up the offset with the list command. This will only work if you've compiled that module with -g to include debugging information.

Send bug reports to dhinds@allegro.stanford.edu. I prefer to handle bug reports by email – please avoid calling me at home or at work.

5.2 Low level PCMCIA debugging aids

The PCMCIA modules contain a lot of conditionally-compiled debugging code. The make.options file shows how to enable this code. A module compiled with PCMCIA_DEBUG set will have a parameter, pc_debug, that controls the verbosity of debugging output. This can be adjusted when the module is loaded, so output can be controlled on a per-module basis without recompiling.

There are a few debugging tools in the debug_tools/ subdirectory of the PCMCIA distribution. The dump_tcic and dump_i365 utilities generate complete register dumps of the PCMCIA controllers, and decode a lot of the register information. They are most useful if you have access to a datasheet for the corresponding controller chip. The dump_tuples utility lists a card's CIS (Card Information Structure), and decodes some of the important bits. And the dump_cisreg utility displays a card's local configuration registers.

The pcmem_cs memory card driver is also sometimes useful for debugging. It can be bound to any PCMCIA card, and does not interfere with other drivers. It can be used to directly access any card's attribute memory or common memory.

5.3 How do I write a Card Services driver for card X?

The Linux PCMCIA Programmer's Guide is the best documentation for the Linux PCMCIA interface. The latest version is always available from cb-iris.stanford.edu in /pub/pcmcia/doc.

For devices that are close relatives of normal ISA devices, you'll probably be able to use parts of existing Linux drivers. In some cases, the biggest stumbling block will be modifying an existing driver so that it can handle adding and removing devices after boot time. Of the current drivers, the memory card driver is the only "self-contained" driver that does not depend on other parts of the Linux kernel to do most of the dirty work.

I've written a skeleton driver with lots of comments that explains a lot of how a driver communicates with Card Services; you'll find this in the PCMCIA source distribution in modules/skeleton.c.