Linux PCMCIA HOWTO

David Hinds, dhinds@allegro.stanford.edu

v1.15, 1995/02/09 07:28:29

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.

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

are included. Specifically, it may be included in commercial distributions without my prior consent. However, I would like to be informed of such usage.

This document may be translated into any language, provided this copyright statement is left intact.

This document is provided "as is", with no explicit or implied warranties. Use the information in this document at your own risk.

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

The current release of Card Services is version 2.4.4.

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, 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.

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 a simple memory card driver that should support most SRAM cards and read-only access to 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.

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.

2 Compilation, installation, and configuration

2.1 Prerequisites and kernel setup

For the latest version, you will need to have kernel version 1.1.89 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/v1.1`, or from `tsx-11.mit.edu` in `/pub/linux/sources/system/v1.1`. Current module utilities can be found in the same places, in the file `modules-1.1.87.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 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.

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`.

2.2 Installation

Unpack the `pcmcia-cs-2.4.4.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`.

Configuration files are kept in the `/etc/pcmcia` directory: do "`make install-etc`" to set up this 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. Finally, "`make install-man`" will install man pages for all the loadable modules and programs.

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.

To use the PCMCIA drivers, first load the core kernel modules:

```
insmod /lib/modules/`uname -r`/pcmcia/pcmcia_core.o
insmod /lib/modules/`uname -r`/pcmcia/i82365.o
insmod /lib/modules/`uname -r`/pcmcia/ds.o
```

Depending on your PCMCIA controller, you may load `tcic.o` instead of `i82365.o`.

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. You should add a line to your system startup file `/etc/rc.d/rc.M` to invoke this:

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

If you are using a PCMCIA ethernet card, you should not try to configure it 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.

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`.

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. Check the corresponding man pages for more details.

2.4 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 2.1 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.

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 get a working network setup, but it is tricky to put everything in the right places by hand. First, with the boot disk mounted on `/mnt`, do:

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

Then, with the root disk mounted on /mnt, 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 cards, and edit `/etc/pcmcia/network` to conform to your network setup.

2.5 When I load the sample `drv_hello.o` module, I get "___moddi3 undefined". What's wrong?

Nothing, really. The `drv_hello` module uses a "modulo" operator that gcc handles by calling a built-in function normally supplied by the libgcc library. Since the module isn't linked against this library, it results in an unresolved reference. Your module utilities are fine.

2.6 Why does `insmod` complain about undefined symbols?

If `pcmcia_core.o` loads fine, but loading `i82365.o` or `tcic.o` fails with undefined symbols like "_check_resource" and "_register_ss_entry", your module utilities (`insmod`, `lsmod`, etc) are out of date. See section 2.1 for more information.

If you see a message like "_kernel_version undefined" or "_init_module undefined" when loading a module that is part of the Linux kernel (like `8390.o`), the object file was not compiled as a loadable module. Make sure you followed all the kernel configuration instructions in section 2.1.

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

The most likely reason is that there is a conflict on the interrupt line being used to signal card status changes. Check `/usr/adm/messages` to see what interrupt is being used by the low level driver (`i82365.o` or `tcic.o`). Unload the PCMCIA modules and re-load this module with a `cs_irq=#` option to select a different value. See the man pages for `i82365` and `tcic` for the lists of valid choices. If you can't find an interrupt number that works, there is also a polled status mode: turn this on with a `poll_interval=100` option to `insmod`, to poll once per second.

3 Usage and features

3.1 How do I tell if it is working?

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 you look when tracking down a problem. When submitting a bug report, you should always include the contents of this file.

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       4     [ds i82365]
```

Your system log file should contain a startup message from `cardmgr`. Inserting a card should generate a series of messages identifying the card and describing how it is configured.

3.2 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/pid/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 `sample.config`.

3.3 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.

The `ibmcc_cs`, `de650_cs`, `3c589_cs`, and `serial_cs` drivers each 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 0x1100 would correspond to irq 8 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`".

3.4 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.

3.5 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.

3.6 How does Card Services deal with suspend/resume?

I've started to integrate APM (Advanced Power Management) support into Card Services. This is working with an internal development version of the APM support package, and should be generally available soon, so stay tuned.

For now, you can do "`cardctl suspend`" before suspending your laptop, and "`cardctl resume`" after resuming, to properly shut down and restart your PCMCIA cards.

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

Use the new `cardctl` 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 3.2 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.
- 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. 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?

This is an old problem that I still have not been able to track down. For some reason, Megahertz modems – specifically, the 2144 model – sometimes fail to get initialized correctly, and get stuck in an unresponsive state. A modem may be incorrectly identified as an "anonymous memory card". If this happens, try using "`cardctl reset #`" to re-initialize the card. If this also fails, try using "`cardctl suspend #`" followed by "`cardctl resume #`", then use the reset command.

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.

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 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.

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.

4.6 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. Unless their policy changes, it is doubtful that Linux drivers for Xircom products will ever become available.

4.7 What's up with support for SCSI adapters?

The Qlogic FastSCSI and New Media Bus Toaster cards now work under Card Services. As of 1.1.81, the Linux kernel supports loadable SCSI driver modules, but you should try to use the latest available kernel.

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.

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`.