Linux Module HOWTO

Lauri Tischler, Editor.
v1.0, 20 June 1996

This is the Module-HOWTO, which is a compilation of all the possible module parameters that can be passed to the Linux modules at load time. Information within is based on Linux Kernel version 2.0.0.

Contents

1 Introduction

Inspiration for this paper came from Paul Gortmaker's excellent BootPrompt-HOWTO. With Paul's permission I have used his paper as a framework, upon which I have grafted my ramblings. Different parts of this paper have been shamelessly, without any permissions what so ever, ripped off from various README and *.txt files within the SourceTree. My sincere thanks for all the authors of those texts which have made this document possible.

This paper generally follows the same structure/module order as the kernel configuration utility `menuconfig` when run in `single_menu_mode=TRUE` mode.

All sections which are marked `Modular` in the configuration utility are included here, regardless whether they have option parameters or not.

Each section follows the example outline below:

```
Load command:
        /sbin/insmod ne.o io=0x300 irq=5

        io = 0          (Explicitly *requires* an "io=0xNNN" value)
        irq = 0         (Tries to determine configured IRQ via autoIRQ)
        (Probes ports: 0x300, 0x280, 0x320, 0x340, 0x360)

        Depends on 8390.o
```

The Module-HOWTO is edited and maintained by:

> Lauri Tischler, `ltischler@efore.fi`

1.1 Disclaimer and Copyright

This document is *not* gospel. However, it is probably the most up to date info that you will be able to find. Nobody is responsible for what happens to your hardware but yourself. If your hardware goes up in smoke (...nearly impossible!) I take no responsibility. ie. THE AUTHOR IS NOT RESPONSIBLE FOR ANY DAMAGES INCURRED DUE TO ACTIONS TAKEN BASED ON THE INFORMATION INCLUDED IN THIS DOCUMENT.

If you are intending to incorporate this document into a published work, please contact me, and I will make an effort to ensure that you have the most up to date information available. In the past, out of date versions of the Linux howto documents have been published, which caused the developers undue grief from being plagued with questions that were already answered in the up to date versions.

1.2 Related Documentation

Following HOWTO's are highly recommended :

- **BootPrompt-HOWTO** by Paul Gortmaker (`Paul.Gortmaker@anu.edu.au`). New versions of this document can be retrieved via anonymous FTP from sunsite.unc.edu, in `/pub/Linux/docs/HOWTO/*` and various Linux ftp mirror sites.
- **kerneld mini-HOWTO** by Henrik Storner (`storner@osiris.ping.dk`). The latest released version of this document can be found at `http://eolicom.olicom.dk/~storner/kerneld-mini-HOWTO.html`. Between releases of the mini-HOWTO you can find updates on Henrik's unstructured list of changes at `http://eolicom.olicom.dk/~storner/kern.html`.

The file `Documentation/Configure.help` gives good general information about what parts of the kernel to modularize or not.

Also plethora of various README files, sprinkled (unfortunately) all over The SourceTree are good and informative reading.

The most up-to-date documentation will always be The Kernel Source itself. For example, if you were looking for what parameters could be passed to the smc-ultra ethernetcard, then you would go to the `linux/drivers/net` directory, and look at the file `smc-ultra.c` – usually somewhere in the end of the file you would find a procedure called `init_module`. Within this procedure or around it you will parameter and structure definitions related to module parameters.

1.3 The Linux Newsgroups

If you have questions about passing parameters to modules at load time, please READ this document first. If this and the related documentation mentioned above does not answer your question(s) then you can try the Linux newsgroups.

General questions on how to configure your system should be directed to comp.os.linux.setup. We ask that you *please* respect this general guideline for content, and don't cross-post your request to other groups.

1.4 New Versions of Document

New versions of this document can be retrieved via anonymous FTP from sunsite.unc.edu, in `/pub/Linux/docs/HOWTO/*` and various Linux ftp mirror sites. Updates will be made as new information or drivers becomes available. If this copy that you are reading is more than 3 months old, it is either out of date, or it means that I have been lazy and haven't updated it.

This document was produced by using the SGML system that was specifically set up for the Linux Howto project, and there are various output formats available, including, postscript, dvi, ascii, html, and soon TeXinfo.

I would recommend viewing it in the html (via a WWW browser) or the Postscript/dvi format. Both of these contain cross-references that are lost in the ascii translation.

If you want to get the official copy off sunsite, here is URL.

Module-HOWTO `<http://sunsite.unc.edu/mdw/HOWTO/Module-HOWTO.html>`

1.5 Experimental (alpha-code) modules or incomplete information.

Here is a list of modules which are declared *experimental* in configuration scripts or otherwise have incomplete info or I'm just too dumb to understand the source code. Pick your choice.

- Support for JAVA binaries.
- Qlogic ISP SCSI driver.
- Frame Relay DLCI driver.
- Sangoma S502A FRAD driver.
- BAYCOM driver for AX.25.
- WIC Radio IP Bridge.
- 3Com 3c505 Ethernet driver.
- 3Com 3c507 Ethernet driver.
- Allied Telesis AT1700 Ethernet driver.
- Intel Ether Express Pro driver.
- Fujitsu FMV-18x Ethernet driver.
- ICL EtherTeam 16i/32 Ethernet driver.
- NI 5210 Ethernet driver.
- Ansel Communications EISA 3200 Ethernet driver.
- Amiga FSS Filesystem.

1.6 History.

- Initial release 1.0.

2 Module utilities

Module utilities are a set of programs necessary for handling the modules. At the time of this writing, version of latest module utilities was `modules-2.0.0.tar.gz`. Info about latest current version can be found at `http://www.pi.se/blox`

This information was originally provided by following gentlemen, Jacques Gelinas `jacques@solucorp.qc.ca` and Björn Ekwall `bj0rn@blox.se` in `/usr/src/linux/Documentation/modules.txt`.

2.1 Making the modules

Your first step is to compile the kernel, as explained in the file `linux/README`. It generally goes like:

- make config
- make dep
- make clean
- make zImage or make zlilo

In `make config`, you select what you want to include in the *resident* kernel and what features you want to have available as loadable modules. You will generally select the minimal resident set that is needed to boot:

- The filesystem of your root partition
- A scsi driver
- Normal hard drive support
- Net support (CONFIG_NET)
- TCP/IP support (CONFIG_INET), but no drivers!
- plus those things that you just can't live without...

The set of modules is constantly increasing, and you will be able to select the option `m` in `make config` for those features that the current kernel can offer as loadable modules.

You also have a possibility to create modules that are less dependent on the kernel version. This option can be selected during `make config`, by enabling `CONFIG_MODVERSIONS`, and is most useful on *stable* kernel versions, such as the kernels from the 1.2 and 2.0 series. If you have modules that are based on sources that are not included in the official kernel sources, you will certainly like this option...

When you have made the kernel, you create the modules by doing:

```
make modules
```

This will compile all modules and update the `linux/modules` directory. In this directory you will then find a bunch of symbolic links, pointing to the various object files in the kernel tree.

Now, after you have created all your modules, you should also do:

```
make modules_install
```

This will copy all newly made modules into subdirectories under `/lib/modules/kernel_release/`, where `kernel_release` is something like 2.0.1, or whatever the current kernel version is...

As soon as you have rebooted the newly made kernel, you can install and remove modules at will with the utilities: `insmod` and `rmmod`. After reading the man-page for insmod, you will also know how easy it is to configure a module when you do `insmod` (hint: symbol=value).

2.2 Extended utilities, `modprobe` and `depmod`.

You also have access to two utilities: `modprobe` and `depmod`, where modprobe is a *wrapper* for (or extension to) `insmod`. These utilities use (and maintain) a set of files that describe all the modules that are available for the current kernel in the `/lib/modules` hierarchy as well as their interdependencies.

Using the modprobe utility, you can load any module like this:

```
/sbin/modprobe module
```

without paying much attention to which kernel you are running, or what other modules this module depends on.

With the help of the modprobe configuration file: `/etc/conf.modules` you can tune the behaviour of modprobe in many ways, including an automatic setting of insmod options for each module. And, yes, there *are* man-pages for all this...

To use modprobe successfully, you generally place the following command in your `/etc/rc.d/rc.S` script. (Read more about this in the `rc.hints` file in the module utilities package, `modules-x.y.z.tar.gz`.)

```
/sbin/depmod -a
```

This computes the dependencies between the different modules. Then if you do, for example

```
/sbin/modprobe umsdos
```

you will automatically load *both* the `msdos` and `umsdos` modules, since `umsdos` runs piggyback on `msdos`.

2.3 The *ultimate* utility, `kerneld`.

OK, you have read all of the above, and feel amply impressed... Now, we tell you to forget all about how to install and remove loadable modules...

With the kerneld daemon, all of these chores will be taken care of automatically. Just answer "Y" to CONFIG_KERNELD in `make config`, and make sure that `/sbin/kerneld` is started as soon as possible after boot and that `/sbin/depmod -a` has been executed for the current kernel. (Read more about this in the module utilities package.)

Whenever a program wants the kernel to use a feature that is only available as a loadable module, and if the kernel hasn't got the module installed yet, the kernel will ask the kerneld deamon to take care of the situation and make the best of it.

This is what happens:

- The kernel notices that a feature is requested that is not resident in the kernel.
- The kernel sends a message to kerneld, with a symbolic description of the requested feature.
- The kerneld daemon asks e.g. modprobe to load a module that fits this symbolic description.

- Modprobe looks into its internal *alias* translation table to see if there is a match. This table can be reconfigured and expanded by having *alias* lines in `/etc/conf.modules`.
- Insmod is then asked to insert the module(s) that modprobe has decided that the kernel needs. Every module will be configured according to the `options` lines in `/etc/conf.modules`.
- Modprobe exits and kerneld tells the kernel that the request succeded (or failed...)
- The kernel uses the freshly installed feature just as if it had been configured into the kernel as a *resident* part.

The icing of the cake is that when an automatically installed module has been unused for a period of time (usually 1 minute), the module will be automatically removed from the kernel as well.

This makes the kernel use the minimal amount of memory at any given time, making it available for more productive use than as just a placeholder for unused code.

Actually, this is only a side-effect from the *real* benefit of kerneld: You only have to create a minimal kernel, that is more or less independent of the actual hardware setup. The setup of the *virtual* kernel is instead controlled by a configuration file as well as the actual usage pattern of the current machine and its kernel.

This should be good news for maintainers of multiple machines as well as for maintainers of distributions.

To use kerneld with the least amount of *hassle*, you need modprobe from a release that can be considered *recent* w.r.t. your kernel, and also a configuration file for modprobe (`/etc/conf.modules`).

Since modprobe already knows about most modules, the minimal configuration file could look something like this:

```
alias scsi_hostadapter aha1542  \# or whatever SCSI adapter you have
alias eth0 3c509                \# or whatever net adapter you have

\# you might need an "options" line for some net adapters:
options 3c509 io=0x300 irq=10

\# you might also need an "options" line for some other module:
options cdu31a cdu31a_port=0x1f88 sony_pas_init=1
```

You could add these lines as well, but they are only *cosmetic*:

```
alias net-pf-3 off      \# no ax25 module available (yet)
alias net-pf-4 off      \# if you don't use the ipx module
alias net-pf-5 off      \# if you don't use the appletalk module
```

Finally, for the *purists*: You can name the modprobe configuration either `/etc/conf.modules` or `/etc/modules.conf`, since modprobe knows what to do in each case...

3 General Kernel setup

Note: you can't have both a.out *and* ELF support compiled as modules. Otherwise, you get a nice Catch-22 when you try to run insmod to install a.out/ELF support so you can run insmod ;-). If you have an all-ELF system, but need a.out for the occasional Netscape session, then you can do a.out support as a module. Otherwise, you should probably leave it in the kernel, and if you haven't gone ELF yet, you can probably leave out ELF compleately.

3.1 Kernel support for a.out binaries (`binfmt_aout.o`).

```
Load command:
        /sbin/insmod binfmt_aout.o

        No module parameters.
```

3.2 Kernel support for ELF binaries (`binfmt_elf.o`).

```
Load command:
        /sbin/insmod binfmt_elf.o

        No module parameters.
```

3.3 Kernel support for JAVA binaries (`binfmt_java.o`).

JAVA is an object oriented programming language developed by SUN; JAVA programs are compiled into "JAVA bytecode" which can then be interpreted by run time systems on many different operating systems. These JAVA binaries are becoming a universal executable format. This option allows you to run a Java binary just like any other Linux program: by typing in its name.

```
Load command:
        /sbin/insmod binfmt_java.o

        No module parameters.
```

4 Floppy and other block devices.

4.1 The Floppy Disk Driver (`floppy.o`).

There are many floppy driver options, and they are all listed in `README.fd` in `linux/drivers/block`. For detailed, up to date, information refer directly to this file.

```
Load command:
        /sbin/insmod floppy.o 'floppy="<options>"'

        Option explanations below..
```

`mask,allowed_drive_mask` - Sets the bitmask of allowed drives to `mask`. By default, only units 0 and 1 of each floppy controller are allowed. This is done because certain non-standard hardware (ASUS PCI motherboards) mess up the keyboard when accessing units 2 or 3. This option is somewhat obsoleted by the cmos option.

`all_drives` - Sets the bitmask of allowed drives to all drives. Use this if you have more than two drives connected to a floppy controller.

`asus_pci` - Sets the bitmask to allow only units 0 and 1. (The default)

`daring` - Tells the floppy driver that you have a well behaved floppy controller. This allows more efficient and smoother operation, but may fail on certain controllers. This may speed up certain operations.

`0,daring` - Tells the floppy driver that your floppy controller should be used with caution.

`one_fdc` - Tells the floppy driver that you have only floppy controller (default)

`two_fdc` *or* `address,two_fdc` - Tells the floppy driver that you have two floppy controllers. The second floppy controller is assumed to be at `address`. This option is not needed if the second controller is at address 0x370, and if you use the 'cmos' option

`thinkpad` - Tells the floppy driver that you have a Thinkpad. Thinkpads use an inverted convention for the disk change line.

`0,thinkpad` - Tells the floppy driver that you don't have a Thinkpad.

`omnibook` *or* `nodma` - Tells the floppy driver not to use Dma for data transfers. This is needed on HP Omnibooks, which don't have a workable DMA channel for the floppy driver. This option is also useful if you frequently get "Unable to allocate DMA memory" messages. Indeed, dma memory needs to be continuous in physical, and is thus harder to find, whereas non-dma buffers may be allocated in virtual memory. However, I advise against this if you have an FDC without a FIFO (8272A or 82072). 82072A and later are OK. You also need at least a 486 to use nodma. If you use nodma mode, I suggest you also set the FIFO threshold to 10 or lower, in order to limit the number of data transfer interrupts.

`dma` - Tells the floppy driver that a workable DMA channel is available (the default).

`nofifo` - Disables the FIFO entirely. This is needed if you get "Bus master arbitration error" messages from your ethernet card (or from other devices) while accessing the floppy.

`fifo` - Enables the FIFO (default)

`[threshold],fifo_depth` - Sets the FIFO threshold. This is mostly relevant in DMA mode. If this is higher, the floppy driver tolerates more interrupt latency, but it triggers more interrupts (i.e. it imposes more load on the rest of the system). If this is lower, the interrupt latency should be lower too (faster processor). The benefit of a lower threshold is less interrupts.

To tune the fifo threshold, switch on over/underrun messages using 'floppycontrol –messages'. Then access a floppy disk. If you get a huge amount of "Over/Underrun - retrying" messages, then the fifo threshold is too low. Try with a higher value, until you only get an occasional Over/Underrun. It is a good idea to compile the floppy driver as a module when doing this tuning. Indeed, it allows to try different fifo values without rebooting the machine for each test. Note that you need to do 'floppycontrol –messages' every time you re-insert the module. Usually, tuning the fifo threshold should not be needed, as the default (0xa) is reasonable.

`[drive],[type],cmos` - Sets the CMOS type of `drive` to `type`. This is mandatory if you have more than two floppy drives (only two can be described in the physical CMOS), or if your BIOS uses non-standard CMOS types. The CMOS types are:

```
          0 - Use the value of the physical CMOS
                1 - 5 1/4 DD
                2 - 5 1/4 HD
                3 - 3 1/2 DD
                4 - 3 1/2 HD
                5 - 3 1/2 ED
                6 - 3 1/2 ED
               16 - unknown or not installed
```

(Note: there are two valid types for ED drives. This is because 5 was initially chosen to represent floppy *tapes*, and 6 for ED drives. AMI ignored this, and used 5 for ED drives. That's why the floppy driver handles both)

`unexpected_interrupts` - Print a warning message when an unexpected interrupt is received (default behavior)

`no_unexpected_interrupts` *or* `L40SX` - Don't print a message when an unexpected interrupt is received. This is needed on IBM L40SX laptops in certain video modes. (There seems to be an interaction between video and floppy. The unexpected interrupt only affect performance, and can safely be ignored.)

4.2 Loopback block device support (`loop.o`).

Enabling this option will allow you to mount a file as a file system. This is useful if you want to check an ISO9660 file system before burning the CD, or want to use floppy images without first writing them to floppy.

```
Load command:
        /sbin/insmod loop.o

        No module parameters.
```

4.3 Multiple devices driver support.

This driver lets you combine several harddisk partitions into one logical block device.

Tools that can be used to manage md devices can be found at `sweet-smoke.ufr-info-p7.ibp.fr/public/Linux/md035.tar.gz`. Same location contains also a document *md-FAQ*.

4.3.1 Multiple device in Linear (append) mode (`linear.o`).

If you use this, then your multiple devices driver will be able to use the so-called linear mode, i.e. it will combine the harddisk partitions by simply appending one to the other.

```
Load command:
        /sbin/insmod linear.o

        No module parameters.
```

4.3.2 Multiple device in RAID-0 (striped) mode (`raid0.o`).

If you use this, then your multiple devices driver will be able to use the so-called raid0 mode, i.e. it will combine the harddisk partitions into one logical device in such a fashion as to fill them up evenly, one chunk here and one chunk there. This will increase the throughput rate if the partitions reside on distinct disks.

```
Load command:
        /sbin/insmod raid0.o

        No module parameters.
```

4.4 RAM disk support (`rd.o`).

Enabling this option will allow you to use a portion of your RAM memory as a block device, so that you can make filesystems on it, read and write to it and do all the other things that normal block devices (such as harddrives) can do. It is usually used to load and store a copy of a minimal root file system off of a floppy into RAM during the initial install of Linux.

```
Load command:
        /sbin/insmod rd.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

4.5 XT harddisk support (xd.o).

Very old 8 bit hard disk controllers used in the IBM XT computer. No, the existence of XT disk support does *NOT* mean that you can run Linux on an IBM XT :).

```
Load command:
        /sbin/insmod xd.o

        No module parameters.
```

5 General TCP/IP and Networking options.

5.1 IP Tunneling (ipip.o and new_tunnel.o).

Tunneling means encapsulating data of one protocol type within another protocol and sending it over a channel that understands the encapsulating protocol. Two modules are necessary **ipip.o**, IP/IP protocol decoder, and **new_tunnel.o** driver, (this was previously called **tunnel.o** and the documentation still refers to this name). Documentation is found at /usr/src/linux/drivers/net/README.tunnel.

```
Load commands:
        /sbin/insmod ipip.o
        /sbin/insmod new_tunnel.o

        No module parameters.
```

5.2 IP Aliasing support (ip_alias.o).

Sometimes it is useful to give several addresses to a single network interface (= serial port or ethernet card). The most common case is that you want to serve different WWW documents to the outside according to which of your host names they used to connect to you.

```
Load command:
        /sbin/insmod ip_alias.o

        No module parameters.
```

5.3 IP Reverse ARP (rarp.o).

```
Load command:
        /sbin/insmod rarp.o

        No module parameters.
```

5.4 IPX Protocol (ipx.o).

This is support for the Novell networking protocol, IPX, commonly used for local networks of DOS and Windows machines.

IPX protocol is required if :

- You wish to connect to Netware servers using Linux DOS emulator *DOSEMU*.
- You wish to mount Netware volumes as Linux filesystems using Linux Novell client *ncpfs*.
- You wish to connect to Linux server from Netware client, or mount Linux directories as Netware volumes. There are two daemons for this purpose, *linwared* and *nwserv*.

Homesite for ncpfs is `ftp.gwdg.de/pub/linux/misc/ncpfs`, but sunsite and its many mirrors will have it as well.

Linware's home site is `klokan.sh.cvut.cz/pub/linux/linware`,

nwserv package *Mars_nwe* is also on `ftp.gwdg.de/pub/linux/misc/ncpfs`.

```
Load command:
        /sbin/insmod ipx.o

        No module parameters.
```

5.5 Appletalk DDP (appletalk.o).

Appletalk is the way Apple computers speak to each other on a network. EtherTalk is the name used for appletalk over ethernet and Localtalk is appletalk over the apple serial links.

```
Load command:
        /sbin/insmod appletalk.o

        No module parameters.
```

6 SCSI Support.

The scsi support in the linux kernel can be modularized in a number of different ways depending upon the needs of the end user. To understand your options, we should first define a few terms.

The scsi-core contains the core of scsi support. Without it you can do nothing with any of the other scsi drivers. The scsi core support can be a module (`scsi_mod.o`), or it can be build into the kernel. If the core is a module, it must be the first scsi module loaded, and if you unload the modules, it will have to be the last one unloaded.

The individual upper and lower level drivers can be loaded in any order once the scsi core is present in the kernel (either compiled in or loaded as a module). The disk driver (`sd_mod.o`), cdrom driver (`sr_mod.o`), tape driver (`st.o`) and scsi generics driver (`sg.o`) represent the upper level drivers to support the various assorted devices which can be controlled. You can for example load the tape driver to use the tape drive, and then unload it once you have no further need for the driver (and release the associated memory).

The lower level drivers are the ones that support the individual cards that are supported for the hardware platform that you are running under. Examples are `aha1542.o` to drive Adaptec 1542 cards.

6.1 SCSI Required Core Support.

6.1.1 SCSI Core support (`scsi_mod.o`).

```
Load command:
        /sbin/insmod scsi_mod.o

        No module parameters.
```

6.2 SCSI High level support.

6.2.1 SCSI Disk support (`sd_mod.o`).

```
Load command:
        /sbin/insmod sd_mod.o

        No module parameters.
```

6.2.2 SCSI Tape support (`st.o`).

```
Load command:
        /sbin/insmod st.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.2.3 SCSI CDrom support (`sr_mod.o`).

```
Load command:
        /sbin/insmod sr_mod.o

        No module parameters.
```

6.2.4 SCSI generic support (`sg.o`).

```
Load command:
        /sbin/insmod sg.o

        No module parameters.
```

6.3 SCSI Low Level Device Driver Support.

Most SCSI-card drivers don't support module parameters, they do generally *autoprobe* for card settings. Do read the SCSI-HOWTO document and README files in `/usr/src/linux/drivers/scsi` to to find out about your hardware. If your card is located in some unconventional i/o-address you must load the driver permanently into the kernel and use Boottime options, see BootPrompt-HOWTO. Optionally you can twiddle the Source and recompile.

6.3.1 7000FASST SCSI support (`wd7000.o`),

```
Load command:
        /sbin/insmod wd7000.o

        No module parameters.
        Autoprobes the card, requires installed BIOS.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.2 Adaptec AHA152X/2825 support (`aha154x.o`).

```
Load command:
        /sbin/insmod aha154x.o

        No module parameters.
        Autoprobes the card, requires installed BIOS.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.3 Adaptec AHA1542 support (`aha1542.o`).

```
Load command:
        /sbin/insmod aha1542.o

        No module parameters.
        Autoprobes card at 0x330 and 0x334 only.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.4 Adaptec AHA1740 EISA support (`aha1740.o`).

```
Load command:
        /sbin/insmod aha1740.o

        No module parameters.
        Autoprobes the card.
```

6.3.5 Adaptec AHA274X/284X/294X support (`aic7xxx.o`).

```
Load command:
        /sbin/insmod aic7xxx.o

        No module parameters.
        Autoprobes the card, BIOS must be enabled.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.6 AdvanSys SCSI support (`advansys.o`).

```
Load command:
        /sbin/insmod advansys.o [parameters]

        asc_iopflag=1                   enable=1, disable=0 port scanning
        asc_ioport=0x110,0x330          ports to scan
        asc_dbglvl=1                    debugging level
                                           0: Errors Only
                                           1: High-Level Tracing
                                           2-N: Verbose Tracing

        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.7 Always IN2000 SCSI support (`in2000.o`).

```
Load command:
        /sbin/insmod in2000.o

        No module parameters.
        Autoprobes the card, BIOS not required.
```

6.3.8 BusLogic SCSI support (`BusLogic.o`).

The list of supported BusLogic cards is *long*. Read `usr/src/linux/drivers/scsi/README.BusLogic` to get the total picture.

```
Load command:
        /sbin/insmod BusLogic.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.9 DTC3180/3280 SCSI support (dtc.o).

```
Load command:
        /sbin/insmod dtc.o

        No module parameters.
        Autoprobes the card.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.10 EATA ISA/EISA (DPT PM2011/021/012/022/122/322) support (eata.o).

```
Load command:
        /sbin/insmod eata.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.11 EATA-DMA (DPT, NEC, AT&T, SNI, AST, Olivetti, Alphatronix) (eata_dma.o).

Includes DPT Smartcache, Smartcache III and SmartRAID.

```
Load command:
        /sbin/insmod eata_dma.o

        No module parameters.
        Autoprobe works in all configurations.
```

6.3.12 EATA-PIO (old DPT PM2001, PM2012A) support (eata_pio.o).

```
Load command:
        /sbin/insmod eata_pio.o

        No module parameters.
```

6.3.13 Future Domain 16xx SCSI support (fdomain.o).

```
Load command:
        /sbin/insmod fdomain.o

        No module parameters.
        Autoprobes the card, requires installed BIOS.
```

6.3.14 Generic NCR5380/53c400 SCSI support (NCR5380.o).

```
Load command:
        /sbin/insmod NCR5380.o

                ncr_irq=xx      the interrupt
                ncr_addr=xx     the port or base address (for port or
                                memory mapped, resp.)
                ncr_dma=xx      the DMA
                ncr_5380=1      to set up for a NCR5380 board
                ncr_53c400=1    to set up for a NCR53C400 board

        modprobe g_NCR5380 ncr_irq=5 ncr_addr=0x350 ncr_5380=1
         - for a port mapped NCR5380 board or

        modprobe g_NCR5380 ncr_irq=255 ncr_addr=0xc8000 ncr_53c400=1
         - for a memory mapped NCR53C400 board with interrupts disabled.

        (255 should be specified for no or DMA interrupt,
         254 to autoprobe for an IRQ line if overridden on the command line.)

        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.15 NCR53c406a SCSI support (NCR53c406a.o).

```
Load command:
        /sbin/insmod NCR53c406a.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.16 NCR53c7,8xx SCSI support (53c7,8xx.o).

```
Load command:
        /sbin/insmod 53c7,8xx.o
```

```
No module parameters.
Autoprobes the card, requires installed BIOS.
Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.17 IOMEGA Parallel Port ZIP drive SCSI support (ppa.o).

Check file `/usr/src/linux/drivers/scsi/README.ppa` for detailed information.

```
Load command:
        /sbin/insmod ppa.o ppa_base=0x378 ppa_nybble=1

        Here are the parameters and their functions:

        Variable        Default    Description

        ppa_base        0x378   The base address of PPA's parallel port.
        ppa_speed_high  1       Microsecond i/o delay used in data transfers
        ppa_speed_low   6       Microsecond delay used in other operations
        ppa_nybble      0       1 to force the driver to use 4-bit mode.
```

6.3.18 PAS16 SCSI support(pas16.o).

```
Load command:
        /sbin/insmod pas16.o

        No module parameters.
        Autoprobes the card, BIOS not required.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.19 Qlogic FAS SCSI support (qlogicfas.o).

```
Load command:
        /sbin/insmod qlogicfas.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.20 Qlogic ISP SCSI support (qlogicisp.o).

Requires firmware.

```
Load command:
        /sbin/insmod qlogicisp.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.21 Seagate ST-02 and Future Domain TMC-8xx SCSI support (seagate.o).

```
Load command:
        /sbin/insmod seagate.o

        No module parameters.
        Autoprobes for address only, irq fixed at 5.
        Requires installed BIOS.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.22 Trantor T128/T128F/T228 SCSI support (t128.o).

```
Load command:
        /sbin/insmod t128.o

        No module parameters.
        Autoprobes the card, requires installed BIOS.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.23 UltraStor 14F/34F support (u14-34f.o).

```
Load command:
        /sbin/insmod u14-34f.o

        No module parameters.
        Autoprobes the card, NOT 0x310 port, BIOS not required.
        Boottime parameters available, see BootPrompt-HOWTO.
```

6.3.24 UltraStor SCSI support (ultrastor.o).

```
Load command:
        /sbin/insmod ultrastor.o

        No module parameters.
        Boottime parameters available, see BootPrompt-HOWTO.
```

7 Network device support

7.1 Required and/or optional lowlevel modules.

7.1.1 Optional BSD compressor for PPP (bsd_comp.o).

```
Load command:
        /sbin/insmod bsd_comp.o

        No module parameters.

        Depends on ppp.o
```

7.1.2 SLHC compressor for PPP (slhc.o).

Routines to compress and uncompress tcp packets (for transmission over low speed serial lines).

Required by SLIP and PPP (also ISDN-PPP) protocols.

```
Load command:
        /sbin/insmod slhc.o

        No module parameters.
```

7.1.3 A general NS8390 ethernet driver core for linux.(8390.o).

This is the chip-specific code for many 8390-based ethernet adaptors. This is not a complete driver, it must be combined with board-specific code such as `ne.o, wd.o, 3c503.o`, etc.

```
Load command:
        /sbin/insmod 8390.o

        No module parameters.
```

7.2 Dummy net driver support (dummy.o).

This is essentially a bit-bucket device (i.e. traffic you send to this device is consigned into oblivion) with a configurable IP address. It is most commonly used in order to make your currently inactive SLIP address seem like a real address for local programs.

```
Load command:
        /sbin/insmod dummy.o

        No module parameters.
```

7.3 EQL (serial line load balancing) support (`eql.o`).

If you have two serial connections to some other computer (this usually requires two modems and two telephone lines) and you use SLIP (= the protocol for sending internet traffic over telephone lines) or PPP (= a better SLIP) on them, you can make them behave like one double speed connection using this driver.

```
Load command:
        /sbin/insmod eql.o

        No module parameters.
```

7.4 Frame relay DLCI support (`dlci.o`).

This is support for the frame relay protocol; frame relay is a fast low-cost way to connect to a remote internet access provider or to form a private wide area network. The one physical line from your box to the local "switch" (i.e. the entry point to the frame relay network) can carry several logical point-to-point connections to other computers connected to the frame relay network. For a general explanation of the protocol, check out `http://frame-relay.indiana.edu/4000/4000index.html` on the WWW. To use frame relay, you need supporting hardware (FRAD) and certain programs from the net-tools package as explained in `Documentation/networking/framerelay.txt`.

```
Load command:
        /sbin/insmod dlci.o

        No module parameters.
```

7.5 Sangoma S502A FRAD support (`sdla.o`).

This is a driver for the Sangoma S502A, S502E and S508 Frame Relay Access Devices. These are multi-protocol cards, but only frame relay is supported by the driver at this time. Please read `Documentation/networking/framerelay.txt`.

```
Load command:
        /sbin/insmod sdla.o

        No module parameters.

        Depends on dlci.o
```

7.6 PLIP (parallel port) support (plip.o).

PLIP (Parallel Line Internet Protocol) is used to create a mini network consisting of two (or, rarely, more) local machines. The parallel ports (the connectors at the computers with 25 holes) are connected using "null printer" or "Turbo Laplink" cables which can transmit 4 bits at a time or using special PLIP cables, to be used on bidirectional parallel ports only, which can transmit 8 bits at a time (you can find the wiring of these cables in `drivers/net/README?.plip`). The cables can be up to 15m long. This works also if one of the machines runs DOS/Windows and has some PLIP software installed, e.g. the Crynwr PLIP packet driver `http://sunsite.cnam.fr/packages/Telnet/PC/msdos/misc/pktdrvr.txt` and winsock or NCSA's telnet.

```
Load command:
        /sbin/insmod plip.o io=0x378 irq=7

        io = 0
        irq = 0         (by default, uses IRQ 5 for port at 0x3bc,
                                      IRQ 7 for port at 0x378,
                                  and IRQ 2 for port at 0x278)
        (Probes ports: 0x278, 0x378, 0x3bc)
```

7.7 PPP (point-to-point) support (ppp.o).

To use PPP you need an additional program called pppd as described in `Documentation/networking/ppp.txt` and in the PPP-HOWTO, available from `sunsite.unc.edu:/pub/Linux/docs/HOWTO`.

```
Load command:
        /sbin/insmod ppp.o

        No module parameters.

        Depends on slhc.o
        Also depends on serial.o, this is not detected automatically,
        so serial.o must be loaded manually.
```

7.8 SLIP (serial line) support (slip.o).

SLIP (Serial Line Internet Protocol) is the protocol used to send Internet traffic over telephone lines or serial cables (also known as nullmodems).

```
Load command:
        /sbin/insmod slip.o
```

```
slip_maxdev = 256 (default value from SL_NRUNIT on slip.h)

Depends on slhc.o
Also depends on serial.o, this is not detected automatically,
so serial.o must be loaded manually.
```

7.9 Amateur Radio and wireless network interfaces.

7.9.1 BAYCOM ser12 and par96 kiss emulation driver for AX.25 (baycom.o).

This is a driver for Baycom style simple amateur radio modems that connect to either a serial interface or a parallel interface. The driver supports the ser12 and par96 designs. To configure the driver, use the setbaycom utility available from http://www.ife.ee.ethz.ch/~sailer/ham/ham.html#lnxbay. For informations on the modems, see `http://www.baycom.de` and `drivers/char/README.baycom`.

```
Load command:
        /sbin/insmod baycom.o modem=1 iobase=0x3f8 irq=4 options=1

        major   major number the driver should use; default 60
        modem   modem type of the first channel (minor 0); 1=ser12,
                2=par96/par97, any other value invalid
        iobase  base address of the port; common values are for ser12 0x3f8,
                0x2f8, 0x3e8, 0x2e8 and for par96/par97 0x378, 0x278, 0x3bc
        irq     interrupt line of the port; common values are for ser12 3,4
                and for par96/par97 7
        options 0=use hardware DCD, 1=use software DCD
```

7.9.2 STRIP (Metricom starmode radio IP) strip.o).

STRIP is a radio protocol developed for the MosquitoNet project (http://mosquitonet.stanford.edu/) to send Internet traffic using Metricom radios. Metricom radios are small, battery powered, 100kbit/sec packet radio transceivers, about the size and weight of a cellular telephone. (You may also have heard them called "Metricom modems" but we avoid the term "modem" because it misleads many people into thinking that you can plug a Metricom modem into a phone line and use it as a modem.) You can use STRIP on any Linux machine with a serial port, although it is obviously most useful for people with laptop computers.

```
Load command:
        /sbin/insmod strip.o

        No module parameters.
```

7.9.3 WaveLAN support (`wavelan.o`).

These are cards for wireless ethernet-like networking. Supported are AT&T GIS and NCR WaveLAN cards.

```
Load command:
        /sbin/insmod wavelan.o

        io = 0x390       (Settable, but change not recommended)
        irq = 0          (Not honoured, if changed..)
```

7.9.4 WIC Radio IP bridge (`wic.o`).

Support for the WIC parallel port radio bridge.

```
Load command:
        /sbin/insmod wic.o

        It appears that devices 'wic0', 'wic1' and 'wic2' have direct
        relations to corresponding 'lpx' ports.
```

7.9.5 Z8530 SCC kiss emulation driver for AX.25 (`scc.o`).

These cards are used to connect your Linux box to an amateur radio in order to communicate with other computers. If you want to use this, read `Documentation/networking/z8530drv.txt` and the HAM-HOWTO.

```
Load command:
        /sbin/insmod scc.o

        No module parameters.
```

7.10 3COM Ethernet cards.

7.10.1 3c501 support (`3c501.o`).

```
Load command:
        /sbin/insmod 3c501.o

        io  = 0x280      IO base address
        irq = 5          IRQ
        (Probes ports:  0x280, 0x300)
```

7.10.2 3c503 support (3c503.o).

```
Load command:
        /sbin/insmod 3c503.o

        io = 0          (It will complain if you don't supply an "io=0xNNN")
        irq = 0         (IRQ software selected by driver using autoIRQ)
        xcvr = 0        (Use xcvr=1 to select external transceiver.)
        (Probes ports: 0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2A0,0x2E0)

        Depends on 8390.o
```

7.10.3 3c505 support (3c505.o).

```
Load command:
        /sbin/insmod 3c505.o

        io = 0x300
        irq = 0
        (Probes ports: 0x300, 0x280, 0x310)
```

7.10.4 3c507 support (3c507.o).

```
Load command:
        /sbin/insmod 3c507.o

        io = 0x300
        irq = 0
        (Probes ports: 0x300, 0x320, 0x340, 0x280)
```

7.10.5 3c509/3c579 support (3c509.o).

```
Load command:
        /sbin/insmod 3c509.o

        io = 0
        irq = 0
        ( Module load-time probing Works reliably only on EISA, ISA ID-PROBE
          IS NOT RELIABLE!  Compile this driver statically into kernel for
          now, if you need it auto-probing on an ISA-bus machine. )
```

7.10.6 3c590 series "Vortex" support (`3c59x.o`).

Provides support for folloving cards :

- 3c590 Vortex 10Mbps.
- 3c595 Vortex 100baseTX.
- 3c595 Vortex 100baseT4.
- 3c595 Vortex 100base-MII.
- EISA Vortex 3c597.

```
Load command:
        /sbin/insmod 3c59x.o debug=1 options=0,,12

        This sets the debug message level to minimal messages,
        sets the first card to the 10baseT transceiver, the second
        to the EEPROM-set transceiver, and the third card to operate
        in full-duplex mode using its 100baseTx transceiver.
        (Note: card ordering is set by the PCI BIOS.)

        Possible media type settings
                0       10baseT
                1       10Mbs AUI
                2       undefined
                3       10base2 (BNC)
                4       100base-TX
                5       100base-FX
                6       MII (not yet available)
                7       <Use default setting>

                8       Full-duplex bit
                8       10baseT full-duplex
                12      100baseTx full-duplex
                16      Bus-master enable bit (experimental use only!)

        Details of the device driver implementation are at the top of
        the source file.
```

7.11 Western Digital/SMC Ethernet cards.

7.11.1 WD80*3 support (`wd.o`).

```
Load command:
```

```
        /sbin/insmod wd.o

        io = 0          (It will complain if you don't supply an "io=0xNNN")
        irq = 0         (IRQ val. read from EEPROM, ancient cards use autoIRQ)
        mem = 0         (Force shared-memory on address 0xC8000, or whatever..)
        mem_end = 0     (Force non-std. mem. size via supplying mem_end val.)
                        (eg. for 32k WD8003EBT, use mem=0xd0000 mem_end=0xd8000
        (Probes ports:  0x300, 0x280, 0x380, 0x240)

        Depends on 8390.o
```

7.11.2 SMC Ultra/EtherEZ support (smc-ultra.o).

```
Load command:
        /sbin/insmod smc-ultra.o

        io = 0          (It will complain if you don't supply an "io=0xNNN")
        irq = 0         (IRQ val. read from EEPROM)
        (Probes ports:  0x200, 0x220, 0x240, 0x280, 0x300, 0x340, 0x380)

        Depends on 8390.o
```

7.11.3 SMC 9194 support (smc9194.o).

This is a driver for SMC's 9000 series of Ethernet cards.

```
Load command:
        /sbin/insmod smc9194.o

        io = 0          for the base address
        irq = 0         for the IRQ
        ifport = 0      for autodetect, 1 for TP, 2 for AUI ( or 10base2 )

        Probes ports:   0x200, 0x220, 0x240, 0x260, 0x280, 0x2A0, 0x2C0, 0x2E0,
                        0x300, 0x320, 0x340, 0x360, 0x380, 0x3A0, 0x3C0, 0x3E0

        Debug level setable in smc9194.c driver code.
```

7.12 Other Ethernet cards.

7.12.1 AT1700 support (at1700.o).

```
Load command:
```

```
/sbin/insmod at1700.o

io = 0x260
irq = 0

(Probes ports: 0x260, 0x280, 0x2A0, 0x240, 0x340, 0x320, 0x380, 0x300)
```

7.12.2 Cabletron E21xx support (e2100.o).

```
Load command:
        /sbin/insmod e2100.o

        io = 0          (It will complain if you don't supply an "io=0xNNN")
        irq = 0         (IRQ software selected by driver)
        mem = 0         (Override default shared memory start of 0xd0000)
        xcvr = 0        (Use xcvr=1 to select external transceiver.)
        (Probes ports: 0x300, 0x280, 0x380, 0x220)

        Depends on 8390.o
```

7.12.3 DEPCA, DE10x, DE200, DE201, DE202, DE422 support (depca.o).

```
Load command:
        /sbin/insmod depca.o

        io = 0x200
        irq = 7
        (Probes ports:  ISA:  0x300, 0x200;
                        EISA: 0x0c00            )
```

7.12.4 EtherWORKS 3 (DE203, DE204, DE205) support (ewrk3.o).

```
Load command:
        /sbin/insmod ewrk3.o

        io = 0x300
        irq = 5
        (With module no autoprobing!
         On EISA-bus does EISA probing.
         Static linkage probes ports on ISA bus:
                0x100, 0x120, 0x140, 0x160, 0x180, 0x1A0, 0x1C0,
                0x200, 0x220, 0x240, 0x260, 0x280, 0x2A0, 0x2C0, 0x2E0,
                0x300,        0x340, 0x360, 0x380, 0x3A0, 0x3C0)
```

7.12.5 EtherExpress 16 support (eexpress.o).

```
Load command:
        /sbin/insmod eexpress.o

        io = 0x300
        irq = 0         (IRQ value read from EEPROM)
        (Probes ports: 0x300, 0x270, 0x320, 0x340)
```

7.12.6 EtherExpressPro support (eepro.o).

```
Load command:
        /sbin/insmod eepro.o

        io = 0x200
        irq = 0
        (Probes ports: 0x200, 0x240, 0x280, 0x2C0, 0x300, 0x320, 0x340, 0x360)
```

7.12.7 Fujitsu FMV-181/182/183/184 support (fmv18x.o).

```
Load command:
        /sbin/insmod fmv18x.o

        io = 0x220      for the base address
        irq = 0         for the IRQ

        Probes ports: 0x220, 0x240, 0x260, 0x280, 0x2a0, 0x2c0, 0x300, 0x340
```

7.12.8 HP PCLAN+ (27247B and 27252A) support (hp-plus.o).

```
Load command:
        /sbin/insmod hp-plus.o

        io = 0          (It will complain if you don't supply an "io=0xNNN")
        irq = 0         (IRQ read from configuration register)
        (Probes ports: 0x200, 0x240, 0x280, 0x2C0, 0x300, 0x320, 0x340)

        Depends on 8390.o
```

7.12.9 HP PCLAN (27245 and other 27xxx series) support (hp.o).

```
Load command:
        /sbin/insmod hp.o
```

```
io = 0          (It will complain if you don't supply an "io=0xNNN")
irq = 0         (IRQ software selected by driver using autoIRQ)
(Probes ports: 0x300, 0x320, 0x340, 0x280, 0x2C0, 0x200, 0x240)

Depends on 8390.o
```

7.12.10 HP 10/100VG PCLAN (ISA, EISA, PCI) support (hp100.o).

```
Load command:
        /sbin/insmod hp100.o

        hp100_port = 0 (IO-base address)

        (Does EISA-probing, if on EISA-slot;
        On ISA-bus probes all ports from 0x100 thru to 0x3E0
        in increments of 0x020)
```

7.12.11 ICL EtherTeam 16i/32 support (eth16i.o).

```
Load command:
        /sbin/insmod eth16i.o

        io = 0x2a0      (It will complain if you don't supply an "io=0xNNN")
        irq = 0         (IRQ software selected by driver using autoIRQ)

        Probed ports on eth16i card :
        0x260, 0x280, 0x2A0, 0x240, 0x340, 0x320, 0x380, 0x300

        Probed ports in eth32i card :
        0x1000, 0x2000, 0x3000, 0x4000, 0x5000, 0x6000, 0x7000, 0x8000,
        0x9000, 0xA000, 0xB000, 0xC000, 0xD000, 0xE000, 0xF000
```

7.12.12 NE2000/NE1000 support (ne.o).

```
Load command:
        /sbin/insmod ne.o

        io = 0          (Explicitly *requires* an "io=0xNNN" value)
        irq = 0         (Tries to determine configured IRQ via autoIRQ)
        (Probes ports: 0x300, 0x280, 0x320, 0x340, 0x360)

        Depends on 8390.o
```

7.12.13 NI5210 support (ni52.o).

```
Load command:
        /sbin/insmod ni52.o

        io=0x360
        irq=9
        memstart=0xd0000
        memend=0xd4000

        Don't use autoprobing, io=0
```

7.13 EISA, VLB, PCI and on board controllers

7.13.1 Ansel Communications EISA 3200 support (ac3200.o).

```
Load command:
        /sbin/insmod ac3200.o

        As this is an EISA board, Autoprobe should work.

        Depends on 8390.o
```

7.13.2 Apricot Xen-II on board ethernet (apricot.o).

```
Load command:
        /sbin/insmod apricot.o

        io = 0x300  (Can't be altered!)
        irq = 10
```

7.13.3 DE425, DE434, DE435, DE450, DE500 support (de4x5.o).

```
Load command:
        /sbin/insmod de4x5.o

        io = 0x000b
        irq = 10
        is_not_dec = 0
        For non-DEC card using DEC 21040/21041/21140 chip, set this to 1
        (EISA, and PCI probing)
```

7.13.4 DECchip Tulip (dc21x4x) PCI support (tulip.o).

```
Load command:
        /sbin/insmod tulip.o

        Read Documentation/networking/tulip.txt
```

7.13.5 Digi Intl. RightSwitch SE-X support (dgrs.o).

This is a Linux driver for the Digi International RightSwitch SE-X EISA and PCI boards. These are 4 (EISA) or 6 (PCI) port ethernet switches and a NIC combined into a single board.

There is a tool for setting up input and output packet filters on each port, called "dgrsfilt".

The management tool lets you watch the performance graphically, as well as set the SNMP agent IP and IPX addresses, IEEE Spanning Tree, and Aging time. These can also be set from the command line when the driver is loaded.

There is also a companion management tool, called "xrightswitch".

```
Load command:
        /sbin/insmod dgrs.o

        debug=NNN               Debug printing level
        dma=0/1                 Disable/Enable DMA on PCI card
        spantree=0/1            Disable/Enable IEEE spanning tree
        hashexpire=NNN          Change address aging time (default 300 seconds)
        ipaddr=A,B,C,D          Set SNMP agent IP address i.e. 199,86,8,221
        ipxnet=NNN              Set SNMP agent IPX network number
```

7.14 Pocket and portable adaptors

7.14.1 D-Link DE600 pocket adaptor support (de600.o).

```
Load command:
        /sbin/insmod de600.o

        de600_debug = 0
        (On port 0x378, irq 7 -- lpt1;  compile time configurable)
```

7.14.2 D-Link DE620 pocket adaptor support (de620.o).

```
Load command:
        /sbin/insmod de620.o
```

```
        bnc = 0, utp = 0        <-- Force media by setting either.
        io = 0x378              (also compile-time configurable)
        irq = 7
```

7.15 Token Ring driver support

7.15.1 Tropic chipset based adaptor support (ibmtr.o).

```
Load command:
        /sbin/insmod ibmtr.o io=0xa20 irq=0

        io = 0xa20      You should not use auto-probing with insmod
        irq = 0
```

7.16 ARCnet support (arcnet.o).

```
Load command:
        /sbin/insmod arcnet.o

        io = 0
        irqnum = 0
        shmem = 0
        num = 0

        DO SET THESE MANUALLY AT INSMOD!

        (When probing, looks at the following possible addresses:
         Suggested ones:
                0x300, 0x2E0, 0x2F0, 0x2D0
         Other ones:
                0x200, 0x210, 0x220, 0x230, 0x240, 0x250, 0x260, 0x270,
                0x280, 0x290, 0x2A0, 0x2B0, 0x2C0,
                       0x310, 0x320, 0x330, 0x340, 0x350, 0x360, 0x370,
                0x380, 0x390, 0x3A0,                      0x3E0, 0x3F0  )
```

8 ISDN subsystem

Setting up ISDN-networking is a complicated task. Read documentation found in /usr/src/linux/Documentation/isdn.

8.1 ISDN support (isdn.o).

```
Load command:
        /sbin/insmod isdn.o

        No module parameters.

        Depends on slhc.o
```

8.2 ICN 2B and 4B support (icn.o).

```
Load command:
        /sbin/insmod icn.o [parameters]

        portbase=p membase=m icn_id=idstring [icn_id2=idstring2]

        p = portbase            (default: 0x320)
        m = shared memory       (default: 0xd0000)

        When using the ICN double card, you MUST define TWO idstrings.
        idstring must start with a character!

        Depends on isdn.o
```

8.3 PCBIT-D support (pcbit.o).

```
Load command:
        /sbin/insmod pcbit.o

        mem = 0         default = 0xd0000
        irq = 0         default = 5

        Depends on isdn.o
```

8.4 Teles/NICCY1016PC/Creatix support (teles.o).

```
Load command:
        /sbin/insmod teles.o io=.....

        io=m0,i0,p0,d0[,m1,i1,p1,d1 ... ,mn,in,pn,dn] teles_id=idstring

        m0 = shared memory of 1st card.         (default: 0xd0000
        i0 = irq of 1st card.                   (default: 15)
        p0 = portbase of 1st card.              (default: 0xd80)
        d0 = D-channel protocol of 1st card.    1=1TR6, 2=EDSS1 (default: 2)
```

```
p1,i1,m1,d1 = Parameters of second card (defaults: none)
pn,in,mn,d1 = Parameters of n'th card (up to 16 cards are supported)

idstring = Driver-Id for accessing with utilities and identification
when using a Line-monitor. (default: none) idstring must start with a character!

The type of the card is determined by the port, irq and shared memory:

port == 0, shared memory != 0   -> Teles S0-8
port != 0, shared memory != 0   -> Teles S0-16.0
port != 0, shared memory == 0   -> Teles S0-16.3

Depends on isdn.o
```

9 CD-ROM drivers (not for SCSI or IDE/ATAPI drives)

9.1 Aztech/Orchid/Okano/Wearnes/TXC/CyDROM support (`aztcd.o`).

```
Load command:
        /sbin/insmod aztcd.o aztcd=[baseaddress]

        Read Documentation/cdrom/aztcd for full information.
```

9.2 Goldstar R420 CDROM support (`gscd.o`).

```
Load command:
        /sbin/insmod gscd.o gscd=[address]

        The default base address is 0x340.
        This will work for most applications.
        Address selection is accomplished by jumpers PN801-1 to PN801-4
        on the GoldStar Interface Card.
        Appropriate settings are: 0x300, 0x310, 0x320, 0x330, 0x340, 0x350,
        0x360, 0x370, 0x380, 0x390, 0x3A0, 0x3B0, 0x3C0, 0x3D0, 0x3E0, 0x3F0
```

9.3 Matsushita/Panasonic/Creative, Longshine, TEAC support (`sbpcd.o`).

```
Load command:
        /sbin/insmod sbpcd.o sbpcd=[address], x

        address = ioaddress
        x       = SBPRO setting, read Documentation/cdrom/sbpcd
```

9.4 Mitsumi (standard) *noXA/Multisession* support (`mcd.o`).

```
Load command:
        /sbin/insmod mcd.o mcd=0x300,11,0x304,5

        Where parameters are io and irq in pairs.
```

9.5 Mitsumi *XA/MultiSession* support (`mcdx.o`).

```
Load command:
        /sbin/insmod mcdx.o mcdx=0x300,11,0x304,5

        Where parameters are io and irq in pairs.
```

9.6 Optics Storage DOLPHIN 8000AT CDROM support (`optcd.o`).

```
Load command:
        /sbin/insmod optcd.o optcd=0x340

        Where parameter is the baseaddress of your card.
```

9.7 Philips/LMS CM206 CDROM support (`cm206.o`).

```
Load command:
        /sbin/insmod cm206.o cm206=0x300,11

        Where parameters are the baseaddress and irq of your card.
```

9.8 Sanyo CDR-H94A CDROM support (`sjcd.o`).

```
Load command:
        /sbin/insmod sjcd.o sjcd_base=0x340

        Where parameter is the baseaddress of your card.
        Default address = 0x340,  no irq,  no dma.
```

9.9 ISP16/MAD16/Mozart soft configurable cdrom interface support (`isp16.o`).

```
Load command:
        /sbin/insmod isp16.o [parameters]
```

```
isp16_cdrom_base=<port>
isp16_cdrom_irq=<irq>
isp16_cdrom_dma=<dma>
isp16_cdrom_type=<drive_type>

Valid values are:
  port=0x340,0x320,0x330,0x360
  irq=0,3,5,7,9,10,11
  dma=0,3,5,6,7
  drive_type=noisp16,Sanyo,Panasonic,Sony,Mitsumi.

  Note that these options are case sensitive.
```

9.10 Sony CDU31A/CDU33A CDROM support (cdu31a.o).

```
Load command:
        /sbin/insmod cdu31a.o [parameters]

        cdu31a_port=<I/O address> - sets the base I/O.
        This must be specified.

        cdu31a_irq=<interrupt> - Sets the interrupt number.
        Leaving this off will turn interrupts off.
```

9.11 Sony CDU535 CDROM support (sonycd535.o).

```
Load command:
        /sbin/insmod sonycd535.o sonycd535=[address]

        Where parameter is the baseaddress of your card.
```

---

10 Filesystems

10.1 Minix fs support(minix.o).

```
Load command:
        /sbin/insmod minix.o

        No module parameters.
```

10.2 Extended fs support (`ext.o`).

```
Load command:
        /sbin/insmod ext.o

        No module parameters.
```

10.3 Second extended fs support (`ext2.o`).

```
Load command:
        /sbin/insmod ext2.o

        No module parameters.
```

10.4 xiafs filesystem support (`xiafs.o`).

```
Load command:
        /sbin/insmod xiafs.o

        No module parameters.
```

10.5 DOS FAT fs support (`fat.o`).

```
Load command:
        /sbin/insmod fat.o

        No module parameters.
```

10.6 MSDOS fs support (`msdos.o`).

```
Load command:
        /sbin/insmod msdos.o

        No module parameters.
        Depends on fat.o.
```

10.7 VFAT (Windows-95) fs support (`vfat.o`).

```
Load command:
        /sbin/insmod vfat.o

        No module parameters.
        Depends on fat.o.
```

10.8 UMSDOS: Unix like fs on top of std MSDOS FAT fs (`umsdos.o`).

```
Load command:
        /sbin/insmod vfat.o

        No module parameters.
        Depends on fat.o and msdos.o.
```

10.9 NFS filesystem support (`nfs.o`).

```
Load command:
        /sbin/insmod nfs.o

        No module parameters.
```

10.10 SMB filesystem support (to mount WfW shares etc..) (`smbfs.o`).

`smbfs` is a filesystem which understands the SMB protocol. This is the protocol Windows for Workgroups, Windows NT or Lan Manager use to talk to each other. `smbfs` was inspired by `samba`, the program written by Andrew Tridgell that turns any unix host into a file server for DOS or Windows clients. See `ftp://nimbus.anu.edu.au/pub/tridge/samba/` for this interesting program suite and lots of more information on SMB and NetBIOS over TCP/IP. There you also find explanation for concepts like netbios name or share.

To use smbfs, you need a special mount program, which can be found in the `ksmbfs` package, found on `sunsite.unc.edu:/pub/Linux/system/Filesystems/smbfs`.

```
Load command:
        /sbin/insmod smbfs.o

        No module parameters.
```

10.11 NCP filesystem support (to mount NetWare volumes) (`ncpfs.o`).

`ncpfs` is a filesystem which understands the NCP protocol, designed by the Novell Corporation for their NetWare(tm) product. NCP is functionally similar to the NFS used in the tcp/ip community. To mount a Netware-Filesystem, you need a special mount program, which can be found in ncpfs package. Homesite for ncpfs is `ftp.gwdg.de/pub/linux/misc/ncpfs`, but sunsite and its many mirrors will have it as well.

Related products are `linware` and `mars_nwe`, which will give Linux partial NetWare Server functionality.

Linware's home site is: `klokan.sh.cvut.cz/pub/linux/linware`,

Mars_nwe can be found on `ftp.gwdg.de/pub/linux/misc/ncpfs`.

```
Load command:
        /sbin/insmod ncpfs.o

        No module parameters.

        Depends on ipx.o
```

10.12 ISO9660 cdrom filesystem support (`isofs.o`).

```
Load command:
        /sbin/insmod isofs.o

        No module parameters.
```

10.13 OS/2 HPFS filesystem support (read only) (`hpfs.o`)

```
Load command:
        /sbin/insmod hpfs.o

        No module parameters.
```

10.14 System V and Coherent filesystem support (`sysv.o`).

This is the implementation of the SystemV/Coherent filesystem for Linux.

It implements all of

- Xenix FS,
- SystemV/386 FS,
- Coherent FS.

```
Load command:
        /sbin/insmod sysv.o

        No module parameters.
```

10.15 Amiga FFS filesystem support (`affs.o`).

```
Load command:
        /sbin/insmod affs.o

        No module parameters.
```

10.16 UFS filesystem support (read only) (`ufs.o`).

Apparently for mounting disks with FreeBSD and/or Sun partitions. No documentation exists, apart from The Source.

```
Load command:
        /sbin/insmod ufs.o

        No module parameters.
```

11 Character devices

11.1 Support for user misc device modules (`misc.o`).

This module is used by **`atixlmouse, busmouse, msbusmouse, psaux, wdt`** and **`softdog`** modules and it is automatically generated if required.

```
Load command:
        /sbin/insmod misc.o

        No module parameters.
```

11.2 Standard/generic serial support (`serial.o`).

NOTE: **`serial.o`** is required by other modules, like ppp.o and slip.o. Also it is required by serial mouse and accordingly by gpm. *However* this dependency is *not* detected by present tools and module **`serial.o`** must be loaded manually.

```
Load command:
        /sbin/insmod serial.o

        No module parameters.
```

11.3 Cyclades async mux support (`cyclades.o`)

```
Load command:
        /sbin/insmod cyclades.o

        No module parameters.
```

11.4 Stallion multiport serial support

The intelligent boards also need to have their "firmware" code downloaded to them. This is done via a user level application supplied in the driver package called "stlload". Compile this program where ever you dropped the package files, by typing "make". In its simplest form you can then type

```
./stlload -i cdk.sys
```

in this directory and that will download board 0 (assuming board 0 is an EasyConnection 8/64 board). To download to an ONboard, Brumby or Stallion do:

```
./stlload -i 2681.sys
```

Read the information at `/usr/src/linux/drivers/char/README.stallion`.

11.4.1 Stallion EasyIO or EC8/32 support (`stallion.o`).

```
Load command:
        /sbin/insmod stallion.o

        No module parameters.
```

11.4.2 Stallion EC8/64, ONboard, Brumby support (`istallion.o`).

```
Load command:
        /sbin/insmod istallion.o

        No module parameters.
```

11.5 SDL RISCom/8 card support (`riscom8.o`).

```
Load command:
        /sbin/insmod riscom8.o [options]

        This driver can support up to 4 boards at time.
        Options : iobase=0xXXX iobase1=0xXXX iobase2=...
```

11.6 Parallel printer support (`lp.o`).

```
Load command:
        /sbin/insmod lp.o io=0x378 irq=0

        io = 0
```

```
irq = 0        irq=0 will run in polled mode

(Probes ports: 0x278, 0x378, 0x3bc)
Note: loading lp.o without any parameters will
grab all parallelports.
```

11.7 Bus Mouse Support

11.7.1 ATIXL busmouse support (atixlmouse.o).

```
Load command:
        /sbin/insmod atixlmouse.o

        No module parameters.
        Depends on misc.o
```

11.7.2 Logitech busmouse support (busmouse.o).

```
Load command:
        /sbin/insmod busmouse.o

        No module parameters.
        Depends on misc.o
```

11.7.3 Microsoft busmouse support (msbusmouse.o).

```
Load command:
        /sbin/insmod msbusmouse.o

        No module parameters.
        Depends on misc.o
```

11.7.4 PS/2 mouse (aka "auxiliary device") support (psaux.o).

```
Load command:
        /sbin/insmod psaux.o

        No module parameters.
        Depends on misc.o
```

11.8 Tape support

For support of SCSI tapedrives, see section *SCSI Support*.

11.8.1 Ftape (QIC-80/Travan) support (ftape.o).

```
Load command:
        /sbin/insmod ftape.o tracing=3

        Option parameter 'tracing' can take following values

        set it to:     to get:
          0             bugs
          1             + errors
          2             + warnings
          3             + information           ** Default **
          4             + more information
          5             + program flow
          6             + fdc/dma info
          7             + data flow
          8             + everything else
```

11.9 Watchdog Timer Support

11.9.1 WDT Watchdog timer (wdt.o).

```
Load command:
        /sbin/insmod wdt.o

        No module parameters.
        Defaults to io=0x240 irq=14,
        make any required changes to wdt.c directly.
        Depends on misc.o
```

11.9.2 Software Watchdog (softdog.o).

```
Load command:
        /sbin/insmod softdog.o

        No module parameters.
        Depends on misc.o
```

12 Sound support (sound.o).

Configuring sound is a complex task, there is a number of Readme-files in directory `/usr/src/linux/drivers/sound`, *read them*.

```
Load command:
        /sbin/insmod sound.o [option]

        Option: dma_buffsize=32768
```

13 Closing

If you have found any glaring typos, or outdated info in this document, please let me know. It is easy to overlook stuff.

Thanks,

Lauri Tischler, ltischler@efore.fi