Linux Kernel Modules Installation HOWTO

rhw@bigfoot.com

Revision History

Revision 1.0	Unknown date	Revised by: rhw
Initial Release		

Describes the installation of Linux kernel modules.

Table of Contents

1. Purpose of this Document

My experience with Linux and modules has been that the existing documents fail to provide a satisfactory explanation as to how to successfully set up Linux with modules configured and working. The procedure explained in this document has been successfully used several times, both on my own system and over the Internet to give directions to somebody trying to get some feature to work which requires a driver supplied only in module form.

My own system runs from a RedHat 4.1 distribution of Linux, and it was on this setup that I developed the procedure. I have since successfully installed it on systems running from various Slackware distributions, and on one system running from a Debian distribution, and the necessary procedure to correctly configure modules under Linux in all three is documented herein.

I have recently used the same procedure with RedHat 4.2, but with inconsistent results on apparently identical systems. I have not yet determined what the problem is, so can make no guarantees at this stage as to whether or not it will work on your system.

---

2. Pre−requisites

- Before the steps in this document can be applied, the reader must have a working Linux installation in which one can get to the Linux prompt as user *root* since the majority of the steps involved can only be undertaken by the said user.
- The existing kernel may be compiled either to use modules or not to use modules, and can even display error messages during the boot−up procedure as a result of modules being configured which aren't available at the moment, providing the above condition is met.
- The source tree for the current kernel is assumed to be found rooted at `/usr/src/linux` and that is also assumed to be the current directory throughout this document at the start of any sequence of commands to be issued.

---

3. Compiler Speed−up

If your machine has 16 or more Megabytes of RAM, there is a useful speed−up that can be done, which is to permit the kernel to compile two or modules in parallel. This will increase the load on the machine whilst the kernel is being recompiled, but will reduce the time during which the compilation will be taking place.

Before you can use this method, you need to check the amount of RAM present in your machine, as if you set this too high, the compilation will actually slow down. Experience has shown that the optimum value depends on the amount of RAM in your system according to the following formula, at least for systems with up to 32 Megabytes of RAM, although it may be a little conservative for systems with larger amounts of RAM:

N = [RAM in Megabytes] / 8 + 1

For the benefit of those with a dislike of math, the values for the common amounts of RAM are as follows:

Table 1. Sample Table

RAM Size	**Value to Use**
16 Megs	3
24 Megs	4
32 Megs	5
40 Megs	6
48 Megs	7
56 Megs	8
64 Megs	9
80 Megs	11
96 Megs	13
112 Megs	15
128 Megs	17

When you have decided on the correct number, edit the file `/usr/src/linux/Makefile` and find the line that currently reads:

```
MAKE=make
```

Replace it with one reading:

```
MAKE=make −j N
```

where `N` is the number determined above.

4. Recompiling the Kernel for Modules

The kernel can be reconfigured to use modules for everything other than the file system mounted as root (in most cases, this is the ext2 file system).

However, there are certain items that appear to be difficult to set up properly as modules, so I would recommend the following be compiled into the kernel:

- Ethernet hardware drivers.
- SCSI CD−ROM drivers.

On the other hand, there are certain driver combinations that *ONLY* work as modules, especially combinations of two or more of the following group:

- A Parallel Printer,
- A Parallel Port drive, such as the *IOMEGA* ZipDrive or JazzDrive, or the *BackPack* CD−ROM drive, and
- The *PLIP* Daemon.

You will need to decide what you are compiling into the kernel, and what as modules, but should take the above points into consideration. The actual choices are made during the compilation, by the second of the following sequence of instructions:

```
cd /usr/src/linux
make menuconfig
make dep clean modules modules_install zImage
```

Having done that, the module dependencies need to be mapped out. This is done with the following command:

```
depmod −a
```

The new kernel now needs to be inserted in the boot chain. I am assuming the reader is using LILO for this purpose, since this is the only loader I have any experience with.

I recommend that one does NOT automatically insert the newly compiled kernel as the default Linux kernel since if it should fail, it is then extremely difficult to recover one's Linux setup without doing a complete reinstallation, which is not to be recommended. For this reason, I have the following entry in my `/etc/lilo.conf` file:

```
image=/usr/src/linux/arch/i386/boot/zImage
  label=new
  alias=n
  read−only
  vga=ask
  optional
```

This entry says that there is an OPTIONAL boot option (which will be ignored if the image in question does not exist) which boots the file `/boot/newlinux` if selected, and allows one to select the video mode it is to be booted in.

Assuming the existence of the above entry in `/etc/lilo.conf` the revised kernel is already correctly located at the end of compilation, and it can be installed via the following command:

```
lilo
```

Having done that, the reader needs to follow the further steps relevant to their selected distribution, as follows:

5.1. Configuring Debian or RedHat for Modules

Prior to carrying out the steps listed here, the steps listed in "*Recompiling the Kernel for Modules*" are assumed to have been carried out.

The Debian and RedHat distributions have identical boot procedures, so also have identical procedures for configuring modules into them.

1. Having logged in as root, use your favourite text editor to create a new file called `/etc/rc.d/init.d/modules.init` with the following contents therein:

   ```
   # Modules initialisation.
   #
   # Start up the module auto-loading daemon.
   /sbin/kerneld

   # Mount all currently unmounted auto-mounted partitions.
   /sbin/mount -a
   ```
2. Having created the above file, perform the following steps whilst logged on as root:

   ```
   cd /etc/rc.d
   chmod 755 init.d/*
   cd rc3.d
   ln -s ../init.d/modules.init 05modules.init
   ```

The system can now be rebooted, and on doing so, it will be found that modules are fully implemented

---

5.2. Configuring Slackware for Modules

Prior to carrying out the steps listed here, the steps listed in "*Recompiling the Kernel for Modules*" are assumed to have been carried out.

The file `/etc/rc.d/rc.M` needs to be edited as follows:

1. Around line 18, there is a section reading as follows:

   ```
   # Screen blanks after 15 minutes idle time.
   /bin/setterm -blank 15
   ```

Immediately after this, insert the following paragraph, with the usual blank lines either side of it:

```
# Load the kernel module auto-loader.
/sbin/kerneld
```

2. About 12 lines further down is the following:

```
# if there is no /etc/HOSTNAME, fall back on this default:
```

Immediately prior to this, insert the following paragraph, again with the usual blank lines either side of it:

```
# Mount remaining unmounted auto-mount drives.
/sbin/mount -a
```

When those changes have been made, save the file.

No further modifications are required for Slackware.

5.3. Configuring Other Distributions for Modules

Prior to carrying out the steps listed here, the steps listed in "*Recompiling the Kernel for Modules*" are assumed to have been carried out.

The precise procedure for other distributions has not been ascertained, but is probably one of the above. To determine which one, display a directory of the contents of the `/etc/rc.d` directory, as follows:

```
cd /etc/rc.d
ls -l *.d rc.*
```

From this resulting display, you can select one of the following three options:

1. If this list includes a directory named `init.d` and some directories with names matching `rc?.d` where the question mark is replaced by single digits, and does *NOT* include a file with the name `rc.M`, that distribution can be configured for modules by following the procedure listed under the title "*Configuring Debian or RedHat for Modules*".
2. If this list does not include a directory named `init.d` but includes a file named `rc.M` then that distribution can be configured for modules by following the procedure listed under the title "*Configuring Slackware for Modules*".
3. If this list matches neither of the above criteria, then the distribution has a boot script not covered by this HowTo. In that case, you are invited to contact the author of this document for advice.