

2

Requirements for Building and Using the Kernel

This chapter describes the programs you need to configure a kernel, build it, and successfully boot it. It's a smart idea to consult the file *Documentation/Changes* to verify the specific version number you should have of each tool described in this chapter. This chapter was based on the 2.6.18 kernel, and describes the versions of tools that work with that kernel. If you are using a different kernel, please verify that you have the required versions as specified in this file, or things might not work properly and it can be very hard to determine what went wrong.

Tools to Build the Kernel



Most Linux distributions offer an installation option to install a range of kernel hacking packages. If your distribution offers this option, it is easiest to install this instead of trying to track down all of the individual programs that are needed for this task.

Only three packages that are needed in order to successfully build a kernel: a compiler, a linker, and a *make* utility. This section describes the contents of each package.

Compiler

The Linux kernel is written in the C programming language, with a small amount of assembly language in some places. To build the kernel, the *gcc* C compiler must be used. Most Linux distributions have a package entitled *gcc* that should be installed. If you wish to download the compiler and build it yourself, you can find it at <http://gcc.gnu.org>.

As of the 2.6.18 kernel release, the 3.2 version of *gcc* is the oldest that can properly build a working kernel. Be warned that getting the most recent *gcc* version is not always a good idea. Some of the newest *gcc* releases don't build the kernel



properly, so unless you wish to help debug compiler bugs, it is not recommended that you try them out.

To determine which version of *gcc* you have on your system, run the following command:

```
$ gcc --version
```

Linker



The C compiler, *gcc*, does not do all of the compiling on its own. It needs an additional set of tools known as *binutils* to do the linking and assembling of source files. The *binutils* package also contains useful utilities that can manipulate object files in lots of useful ways, such as to view the contents of a library.

binutils can usually be found in a distribution package called (not surprisingly) *binutils*. If you wish to download and install the package yourself, you can find it at <http://www.gnu.org/software/binutils>.

As of the 2.6.18 kernel release, the 2.12 release of *binutils* is the oldest that can successfully link the kernel. To determine which version of *binutils* you have on your system, run the following command:

```
$ ld -v
```

make



make is a tool that walks the kernel source tree to determine which files need to be compiled, and then calls the compiler and other build tools to do the work in building the kernel. The kernel requires the GNU version of *make*, which can usually be found in a package called *make* for your distribution.

If you wish to download and install *make* yourself, you can find it at <http://www.gnu.org/software/make>.

As of the 2.6.18 kernel release, the 3.79.1 release of *make* is the oldest that can properly build the kernel. It is recommended that you install the latest stable version of *make*, because newer versions are known to work faster at processing the build files.

To determine which version of *make* you have on your system, run the following command:

```
$ make --version
```

Tools to Use the Kernel

While the version of the kernel that is running does not usually affect any user application, there are a small number of program for which the kernel version is important. This section describes a number of tools that are probably already installed on your Linux system. If you upgrade your kernel to a version different from the one that came with your distribution, some of these packages may also need to be upgraded in order for the system to work properly.



util-linux

The *util-linux* package is a collection of small utilities that do a wide range of different tasks. Most of these utilities handle the mounting and creation of disk partitions and manipulation of the hardware clock in the system.

If you wish to download and install the *util-linux* package yourself, you can find it at <http://www.kernel.org/pub/linux/utils/util-linux>.

As of the 2.6.18 kernel release, the 2.10 release of *util-linux* is the oldest that works properly. It is recommended that you install the latest version of this package, because new version support new features added to the kernel. Bind mounts are one example of an option in newer kernels, and a newer version of *util-linux* is needed in order to have them work properly.

To determine which version of the *util-linux* package you have on your system, run the following command:

```
$ fdformat --version
```

module-init-tools

The *module-init-tools* package is needed if you wish to use Linux kernel modules. A *kernel module* is a loadable chunk of code that can be added to or removed from the kernel while the kernel is running. It is useful to compile device drivers as modules and then load only the ones that correspond to the hardware present in the system. All Linux distributions use modules in order to load only the needed drivers and options for the system based on the hardware present, instead of being forced to build all possible drivers and options in the kernel in one large chunk. Modules save memory by loading just the code that is needed to control the machine properly.

The kernel module loading process underwent a radical change in the 2.6 kernel release. The linker for the module (the code that resolves all symbols and figures out how to put the pieces together in memory) is now built into the kernel, which makes the userspace tools quite small. Older distributions have a package called *modutils* that does not work properly with the 2.6 kernel. The *module-init-tools* package is what you need to get the 2.6 kernel to work properly with modules.

If you wish to download and install the *module-init-tools* package yourself, you can find it at <http://www.kernel.org/pub/linux/utils/kernel/module-init-tools>.

As of the 2.6.18 kernel release, the 0.9.10 release of *module-init-tools* is the oldest version that works properly. It is recommended that the latest version of this package be installed, as new features added to the kernel can be used by newer versions of this package. Blacklisting modules to prevent them from being automatically loaded by the *udev* package is one such option that is present in newer versions of *module-init-tools*, but not older ones.

To determine which version of the *module-init-tools* package you have on your system, run the following command:

```
$ depmod -V
```

Filesystem-Specific Tools

A wide range of tools specific to particular filesystems are necessary to create, format, configure, and fix disk partitions. The *util-linux* package has a few of these utilities, but some of the more popular filesystems have separate packages that contain the necessary programs.

ext2/ext3/ext4

The *ext3* and experimental *ext4* filesystems are upgrades of *ext2* and can be managed with the same tools; any recent version of an *ext2*-based tool can work with the other two filesystems as well.

To work with any of these filesystems, you must have the *e2fsprogs* package. If you wish to download and install this package yourself, you can find it at <http://e2fsprogs.sourceforge.net>.

As of the 2.6.18 kernel release, the 1.29 release of *e2fsprogs* is the oldest that works properly with the kernel. It is highly recommended that you use the newest version in order to take advantage of newer features in the *ext3* and *ext4* filesystems.

To determine which version of *e2fsprogs* you have on your system, run the following command:

```
$ tune2fs
```

JFS

To use the JFS filesystem from IBM, you must have the *jfsutils* package. If you wish to download and install this package yourself, you can find it at <http://jfs.sourceforge.net>.

As of the 2.6.18 kernel release, the 1.1.3 release of *jfsutils* is the oldest that works properly with the kernel. To determine which version of *jfsutils* you have on your system, run the following command:

```
$ fsck.jfs -V
```

ReiserFS

To use the ReiserFS filesystem, you must have the *reiserfsprogs* package. If you wish to download and install this package yourself, you can find it at <http://www.namesys.com/download.html>.

As of the 2.6.18 kernel release, the 3.6.3 release of *reiserfsprogs* is the oldest that works properly with the kernel. To determine which version of *reiserfsprogs* you have on your system, run the following command:

```
$ reiserfsck -V
```

XFS

To use the XFS filesystem from SGI, you must have the *xfsprogs* package. If you wish to download and install this package yourself, you can find it at <http://oss.sgi.com/projects/xfs>.

As of the 2.6.18 kernel release, the 2.6.0 release of *xfsprogs* is the oldest that works properly with the kernel. To determine which version of *xfsprogs* you have on your system, run the following command:

```
$ xfs_db -V
```

Quotas

To use the quota functionality of the kernel, you must have the *quota-tools* package.* This package includes programs that let you set quotas on users, provide statistics on the amount of quota being used by different users, and issue warnings when people get too close to using up their available filesystem quota.

If you wish to download and install this package yourself, you can find it at <http://sourceforge.net/projects/linuxquota>.

As of the 2.6.18 kernel release, the 3.09 release of *quota-tools* is the oldest that works properly with the kernel. To determine which version of *quota-tools* you have on your system, run the following command:

```
$ quota -V
```

NFS

To use the NFS filesystem properly, you must have the *nfs-utils* package.† This package includes programs that let you mount NFS partitions as a client, and run an NFS server.

If you wish to download and install this package yourself, you can find it at <http://nfs.sf.net>.

As of the 2.6.18 kernel release, the 1.0.5 release of *nfs-utils* is the oldest that works properly with the kernel. To determine which version of *nfs-utils* you have on your system, run the following command:

```
$ showmount --version
```

Other Tools

There are a few other important programs that are closely tied to the kernel version. These programs are not usually required in order for the kernel to work properly, but they enable access to different types of hardware and functions.

* Some distributions, notably Debian, call this package *quota* instead of *quota-tools*.

† Some distributions, notably Debian, call this package *nfs-common* instead of *nfs-utils*.

udev

udev is a program that enables Linux to provide a persistent device-naming system in the */dev* directory. It also provides a dynamic */dev*, much like the one provided by the older (and now removed) *devfs* filesystem. Almost all Linux distributions use *udev* to manage the */dev* directory, so it is required in order to properly boot the machine.

Unfortunately, *udev* relies on the structure of */sys*, which has been known to change from time to time with kernel releases. Some of these changes in the past have been known to break *udev*, so that your machine will not boot properly. If you have the latest version of *udev* recommended for your kernel and have problems with it working properly, please contact the *udev* developers on the mailing list available at linux-hotplug-devel@lists.sourceforge.net.

It is highly recommended that you use the version of *udev* that comes with your Linux distribution, as it is tied into the distribution specific boot process very tightly. But if you wish to upgrade *udev* on your own, you can find it at <http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html>.

As of the 2.6.18 kernel release, the 081 release of *udev* is the oldest that works properly with the kernel. It is recommended that you use the latest version of *udev*, because it will work better with newer kernels, due to changes in how *udev* and the kernel communicate.

To determine which version of *udev* you have on your system, run the following command:

```
$ udevinfo -V
```

Process tools

The package *procps* includes the commonly used tools *ps* and *top*, as well as many other handy tools for managing and monitoring processes running on the system.

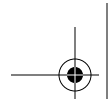
If you wish to download and install this package yourself, you can find it at <http://procps.sourceforge.net>.

As of the 2.6.18 kernel release, the 3.2.0 release of *procps* is the oldest that works properly with the kernel. To determine which version of *procps* you have on your system, run the following command:

```
$ ps --version
```

PCMCIA tools

In order to properly use PCMCIA devices with Linux, a userspace helper program must be used to set up the devices. For older kernel versions, this program was called *pcmcia-cs*, but that has been replaced with a much simpler system called *pcmciautils*. If you wish to use PCMCIA devices, you must have this package installed for them to work properly.



If you wish to download and install this package yourself, you can find it at <ftp://ftp.kernel.org/pub/linux/utils/kernel/pcmcia>.

As of the 2.6.18 kernel release, the 004 release of *pcmciautils* is the oldest that works properly with the kernel. But the latest version is recommended in order to take advantage of newer features in the PCMCIA subsystem, such as automatic driver loading when new devices are found.

To determine which version of *pcmciautils* you have on your system, run the following command:

```
$ pccardctl -V
```

Requirements

