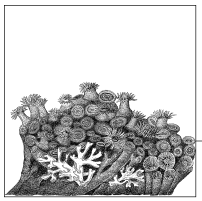

8
Kernel Configuration Recipes

Previous chapters taught the mechanics of reconfiguring the kernel; the payoff comes in this chapter where you can find all the most common kinds of changes people need to make to their kernels, with specific instructions on how to do so.

Disks

The Linux kernel supports a wide range of different disk types. This section shows how to configure the kernel so that it supports most of the more common types of disk controllers.

USB Storage

To use a USB storage device (commonly referred to as USB "flash" device, or an external USB disk drive) USB support must be first working properly. Refer to the recipe in the section called "USB" for how to do this.

A USB storage device can be identified by using the *lsusb* program. If the following command sequence produces the results shown, a USB storage device is present on the system:

```
$ /usr/sbin/lsusb -v | grep Storage
      bInterfaceClass         8 Mass Storage
```

Enable it as follows.

1. A USB Storage device is in reality a USB SCSI device that talks over a USB connection. Because of this, the SCSI subsystem must be enabled:

```
Device Drivers
    SCSI Device Support
        [*] SCSI Device Support
```

2. Also in the SCSI system, the "SCSI disk support" must be enabled in order for the device to be mounted properly:

```
Device Drivers
    SCSI Device Support
        [*] SCSI disk support
```

3. Enable USB Storage support:

```
Device Drivers
   USB Support
   [M] USB Mass Storage support
```

A number of specific USB storage devices are listed as separate configuration items, as they do not follow the standard USB specification and require special code. If you have one of these devices, please enable support for them.

IDE Disks

IDE disks are the most common type of PC disks. The device that enables them to work properly is an IDE disk controller. To determine whether you have a IDE disk controller on the system, use the *lspci* command in the following manner:*

```
$ /usr/sbin/lspci | grep IDE
00:1f.1 IDE interface: Intel Corporation 82801EB/ER (ICH5/ICH5R) IDE
Controller (rev 02)
00:1f.2 IDE interface: Intel Corporation 82801EB (ICH5) SATA Controller (rev
02)
```

Note that your response will probably not be identical; what is important is that the command shows some an IDE controller (the first device in the previous example.) If you find only SATA controllers, please see the next section "Serial ATA (SATA)." Now perform the following steps.

1. Enable PCI support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
```

2. Enable the IDE subsystem, and IDE support:

```
Device Drivers
    [*] ATA/ATAPI/MFM/RLL support
    [*]   Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
```

3. In the ATA system, the specific type of IDE controller that you have must be enabled in order for it to work properly. To provide a good backup in case you choose the wrong type, select the "generic" IDE controller:

```
Device Drivers
    ATA/ATAPI/MFM/RLL support
        [*]     generic/default IDE chipset support
```

* Almost all distributions place the *lspci* program in the */usr/sbin/* directory, but some place it in other locations. To find out where it is located, enter:

```
$ which lspci
/usr/sbin/lspci
```

If you are using a distribution that puts it somewhere else, please use that path for whenever we discuss using *lspci*.

4. Enable the different PCI IDE controllers:

```
Device Drivers
    ATA/ATAPI/MFM/RLL support
        [*]     PCI IDE chipset support
```

This opens up a lengthy submenu of the different IDE controller types. Select the proper one based on the name of the device you found in the *lspci* step.

Serial ATA (SATA)

SATA is a type of disk controller that is the successor to the IDE disk controller. To determine if you have a SATA disk controller on the system, run the following command:

```
$ /usr/sbin/lspci | grep SATA
00:1f.2 IDE interface: Intel Corporation 82801EB (ICH5) SATA Controller (rev
02)
```

Note that your response will probably not be identical; what is important is that the command shows some SATA devices.

SATA disks use a kernel library called *libata* that handles most of the SATA-specific functionality. That library uses the SCSI layer to talk to the block layer, so several different kernel options need to be enabled in order for SATA disks to work properly.

1. Enable PCI support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
```

2. Enable the SCSI subsystem:

```
Device Drivers
    SCSI Device Support
        [*] SCSI Device Support
```

3. Also in the SCSI system, the `SCSI disk support` option must be enabled in order for the device to be mounted properly:

```
Device Drivers
    SCSI Device Support
        [*] SCSI disk support
```

4. The SATA options are under the "SCSI low-level drivers" section:

```
Device Drivers
    SCSI Device Support
        SCSI low-level drivers
            [*] Serial ATA (SATA) support
```

5. In that section, enable the specific SATA controller type that you have. Look at the output of the previously mentioned *lspci* command for a list of the types of SATA controllers that are present on your system. For example, most motherboards from Intel require the PIIX/ICH SATA driver (as the previous example showed):

```
Device Drivers
    SCSI Device Support
        SCSI low-level drivers
```

```
[*] Serial ATA (SATA) support
[*]   Intel PIIX/ICH SATA support
```

Burning a CD-ROM

Burning a CD-ROM is very simple on Linux. If your kernel can support reading from a CD-ROM, it can also support burning a CD-ROM. There are two ways to enable CD-ROM support in Linux, one for IDE drives and one for SCSI and SATA drives.

IDE CD-ROM drives

IDE CD-ROM drives are controlled by the same IDE controller as your main IDE disk drives. Make sure the IDE controller is properly supported as described earlier in "IDE Disks." If it is properly supported, only one other configuration item needs to be selected:

```
Device Drivers
    [*] ATA/ATAPI/MFM/RLL support
    [*]   Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
    [M]     Include IDE/ATAPI CDROM support
```

SCSI and SATA CD-ROM drives

SATA and SCSI CD-ROM drives are controlled by the same controller as your main disk drives. Make sure the SATA or SCSI controller is properly supported. For SATA disks, see the earlier section "Serial ATA (SATA)."

To support SATA or SCSI CD-ROM drives, the SCSI CD-ROM driver must be enabled:

```
Device Drivers
    SCSI Device Support
        [*] SCSI CDROM support
```

Once that is enabled, the SATA or SCSI CD-ROM drive should work properly.

Devices

Linux supports a vast range of different types of devices (more than any other operating system ever has). This section shows how to enable some of the more common types.

USB

Linux supports many different types of USB devices. To enable USB support, you must first enable support for a USB controller, which drives the USB connection on the machine.

To determine if your machine has a USB controller, and which type it is, run the following command:

```
$ /usr/sbin/lspci | grep USB
00:1d.0 USB Controller: Intel Corporation 82801EB/ER (ICH5/ICH5R) USB UHCI
Controller #1 (rev 02)
00:1d.1 USB Controller: Intel Corporation 82801EB/ER (ICH5/ICH5R) USB UHCI
Controller #2 (rev 02)
00:1d.2 USB Controller: Intel Corporation 82801EB/ER (ICH5/ICH5R) USB UHCI
Controller #3 (rev 02)
00:1d.3 USB Controller: Intel Corporation 82801EB/ER (ICH5/ICH5R) USB UHCI
Controller #4 (rev 02)
00:1d.7 USB Controller: Intel Corporation 82801EB/ER (ICH5/ICH5R) USB2 EHCI
Controller (rev 02)
```

Note that your response will probably not be identical; what is important is that the command shows some USB controllers.

1. Enable PCI support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
```

2. Enable USB support for the kernel:

```
Device Drivers
    USB Support
        [M] Support for Host-side USB
```

3. Enable the specific USB Host controllers for your machine (it is safe to enable them all if you do not know which you have):

```
Device Drivers
    USB Support
    ---   USB Host Controller Drivers
    [M]   EHCI HCD (USB 2.0) support
    [M]   OHCI HCD support
    [M]   UHCI HCD (most Intel and VIA) support
```

4. Individual USB devices also need their drivers to be enabled. A large majority of them are under the main USB driver section:

```
Device Drivers
    USB Support
```

   But some devices, such as USB video and DVB and sound, are listed in the section controlling all of these types of devices. For example, the USB sound driver can be found under the Sound menu:

```
Device drivers
    Sound
        [*] Sound card support
            [*] Advanced Linux Sound Architecture
                USB Devices
                    [M] USB Audio/MIDI driver
```

If you want to insert USB storage devices (USB flash), look now at the section called "USB Storage," at the beginning of this chapter.

IEEE 1394 (FireWire)

IEEE 1394 is commonly known by the name FireWire, the name by which Apple Computer publicized it. IEEE 1394 is a high-speed bus that connects external devices, much as USB does.

To determine whether your machine has a FireWire controller and which type it is, run the following command:

```
$ /usr/sbin/lspci | grep FireWire
06:0c.0 FireWire (IEEE 1394): Texas Instruments TSB43AB22/A IEEE-1394a-2000
Controller (PHY/Link)
06:0d.2 FireWire (IEEE 1394): Creative Labs SB Audigy FireWire Port (rev 04)
```

Note that your response will probably not be identical; what is important is that the command shows some FireWire controllers.

1. Enable PCI support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
```

2. Enable IEEE 1394 support for the kernel:

```
Device Drivers
    IEEE 1394 (FireWire) support
        [*] IEEE 1394 (FireWire) support
```

3. Enable the specific type of FireWire host controller you have:

```
Device Drivers
    IEEE 1394 (FireWire) support
        [*] IEEE 1394 (FireWire) support
        ---   Device Drivers
        [M]   Texas Instruments PCILynx support
        [M]   OHCI-1394 support
```

4. Finally, enable the specific type of FireWire devices you have:

```
Device Drivers
    IEEE 1394 (FireWire) support
        [*] IEEE 1394 (FireWire) support
        ---   Protocol Drivers
        [M]   OHCI-1394 Video support
        [M]   SBP-2 support (Harddisks etc.)
        [ ]     Enable Phys DMA support for SBP2 (Debug)
        [M]   Ethernet over 1394
        [M]   OHCI-DV I/O support
        [M]   Raw IEEE1394 I/O support
```

PCI Hotplug

PCI hotplug systems are becoming more popular with the use of ExpressCard and laptop docking stations.

To determine whether your machine has an ExpressCard controller, look at the hardware to see whether an ExpressCard card can be plugged into it.

1. Enable PCI support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
```

2. Enable PCI hotplug support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
    PCI Hotplug Support
        [M] Support for PCI Hotplug (EXPERIMENTAL)
```

3. There is a wide range of different types of PCI hotplug controllers. For most laptops and for ExpressCard support, enable the ACPI controller:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
    PCI Hotplug Support
        [M] Support for PCI Hotplug (EXPERIMENTAL)
        [M]   ACPI PCI Hotplug driver
```

4. Also enable the PCI Express controller:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
    [*] PCI Express Support
    [M]   PCI Express Hotplug driver
```

PCMCIA/CardBus

PCMCIA and CardBus device support is in almost every laptop manufactured. Newer laptops, however, are switching to the ExpressCard format (see the PCI Hotplug recipe in the previous section, "PCI Hotplug").

To determine whether your machine has a PCMCIA controller, look at the hardware to see whether a PCMCIA card can be plugged into it.

1. Enable PCI support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    [*] PCI Support
```

2. Enable PCCARD support for the kernel:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    PCCARD (PCMCIA/CardBus) support
        [M] PCCard (PCMCIA/CardBus) support
```

3. Enable both PCMCIA and CardBus support to cover the widest range of devices:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    PCCARD (PCMCIA/CardBus) support
        [M] PCCard (PCMCIA/CardBus) support
        [M]   16-bit PCMCIA support
        [*]   32-bit CardBus support
```

   Enable the card bridge type for your laptop. The most common one is the "yenta-like" controller:

```
Bus options (PCI, PCMCIA, EISA, MCA, ISA)
    PCCARD (PCMCIA/CardBus) support
        [M] PCCard (PCMCIA/CardBus) support
```

```
[M]   CardBus yenta-compatible bridge support
[ ]   Cirrus PD6729 compatible bridge support
[ ]   i82092 compatible bridge support
[ ]   i82365 compatible bridge support
[ ]   Databook TCIC host bridge support
```

Sound (ALSA)

Advanced Linux Sound Architecture (ALSA) is the current sound system for the Linux kernel. An earlier sound system (OSS) has been deprecated, and almost all of the older drivers have been removed from the kernel source tree.

To determine which type of sound controller is present in your machine, and what type it is, run the following command:

```
$ /usr/sbin/lspci | grep -i audio
00:1f.5 Multimedia audio controller: Intel Corporation 82801EB/ER (ICH5/
ICH5R) AC'97 Audio Controller (rev 02)
06:0d.0 Multimedia audio controller: Creative Labs SB Audigy (rev 04)
```

Note that your response will probably not be identical; what is important is that the command shows some Audio controllers.

1. Enable basic sound support:

```
Device Drivers
    Sound
        [M] Sound Card Support
```

2. Enable ALSA:

```
Device Drivers
    Sound
        [M] Sound Card Support
            [M] Advanced Linux Sound Architecture
```

3. There are a number of different base ALSA options, such as support for the older OSS sound protocol. If you have older applications, you should enable the related options:

```
Device Drivers
    Sound
        [M] Sound Card Support
            [M] Advanced Linux Sound Architecture
            [M]   OSS Mixer API
            [M]   OSS PCM (digital audio) API
            [ ]     OSS PCM (digital audio) API - Include plugin system
```

4. Enable the specific type of sound device that you have. PCI sound cards are under the PCI submenu:

```
Device Drivers
    Sound
        [M] Sound Card Support
            [M] Advanced Linux Sound Architecture
                PCI Devices
```

CPU

If you wish to have the Linux kernel run as fast as possible for your specific processor and hardware type, there are a few options that you can set to get the last bit of performance out of the hardware. This section will show some of the different processor-specific options that you can tune for your processor.

Processor Types

A wide range of specific processor options are available to be changed in the Linux kernel. The most important one for our purpose specifies the exact type of CPU you are using this kernel for. To determine the type of processor you are using, run the following command:

```
$ cat /proc/cpuinfo  | grep "model name"
model name      : Intel(R) Xeon(TM) CPU 3.20GHz
```

Note that your response will probably not be identical; what is important is that the command shows the model name of the processor present on the system.

1. Select the subarchitecture type of the processor:

```
Processor type and features
    Subarchitecture Type
        (X) PC-compatible
        ( ) AMD Elan
        ( ) Voyager (NCR)
        ( ) NUMAQ (IBM/Sequent)
        ( ) Summit/EXA (IBM x440)
        ( ) Support for other sub-arch SMP systems with more than 8 CPUs
        ( ) SGI 320/540 (Visual Workstation)
        ( ) Generic architecture (Summit, bigsmp, ES7000, default)
        ( ) Support for Unisys ES7000 IA32 series
```

   Only if your machine is one of the other types in the preceding list should you select anything other than the `PC-compatible` option. However, if you wish to create a single kernel that will run on all of the types of machines shown, select the `Generic architecture` option. Some of the above options might not be present if you have not also selected the `Symmetric multi-processing support` option.

2. Select the processor family type. The `PC-compatible` option needs to be selected from the previous options for this submenu to be displayed:

```
Processor type and features
    Processor family
        ( ) 386
        ( ) 486
        ( ) 586/K5/5x86/6x86/6x86MX
        ( ) Pentium-Classic
        ( ) Pentium-MMX
        ( ) Pentium-Pro
        ( ) Pentium-II/Celeron(pre-Coppermine)
        ( ) Pentium-III/Celeron(Coppermine)/Pentium-III Xeon
        ( ) Pentium M
        (X) Pentium-4/Celeron(P4-based)/Pentium-4 M/Xeon
```

```
( ) K6/K6-II/K6-III
( ) Athlon/Duron/K7
( ) Opteron/Athlon64/Hammer/K8
( ) Crusoe
( ) Efficeon
( ) Winchip-C6
( ) Winchip-2
( ) Winchip-2A/Winchip-3
( ) GeodeGX1
( ) Geode GX/LX
( ) CyrixIII/VIA-C3
( ) VIA C3-2 (Nehemiah)
( ) Generic x86 support
```

For more details on this configuration item, please refer to the entry for M386 in Chapter 11 for a full description of how to pick the proper processor type depending on what processor you have, and what range of machines you wish the kernel to run on.

SMP

If your system contains more than one CPU, or a Hyperthreaded or Dual Core CPU, you should select the multiprocessor option for the Linux kernel in order to take advantage of the additional processors. Unless you do, you will be wasting the other processors by not using them at all.

Enable multiprocessing:

```
Processor type and features
    [*] Symmetric multi-processing support
```

Preemption

Systems running as servers have very different workload requirements from those being used as a desktop for video and audio applications. The kernel allows different modes of "preemption" in order to handle these different workloads. *Preemption* is the ability of the kernel to interrupt itself while it is doing something else, in order to work on something with a higher priority, such as updating a sound or video program.

To change to a different preemption model, use this menu:

```
Processor type and features
    Preemption Model
        (X) No Forced Preemption (Server)
    ( ) Voluntary Kernel Preemption (Desktop)
    ( ) Preemptible Kernel (Low-Latency Desktop)
```

If you wish to make the kernel even more responsive to higher priority tasks than the general preemption option provides, you can also allow interruptions to one of the main internal kernel locks:

```
Processor type and features
    [*] Preempt The Big Kernel Lock
```

This option is able to be selected only if you have already selected either the `Preemptible Kernel` or `Symmetric multi-processing support` options.

Suspend

The Linux kernel has the ability to suspend itself to disk, allowing you to disconnect the power, and then at a later time, power up and resume exactly where the machine was when it was suspended. This functionality is very useful on laptops that run Linux.

Enable this by selecting:

```
Power management options (ACPI, APM)
    [*] Software Suspend
```

The kernel needs to know where to save the suspended kernel image to, and then later where to resume it from. This location is usually a kernel swap partition on the disk. To specify which partition this should be set:

```
Power management options (ACPI, APM)
    (/dev/hda3) Default resume partition
```

Make sure you specify the proper partition to suspend the machine to, and do not use a partition that is being used by the system for data. The proper partition name can be found by running the following command:

```
$ /sbin/swapon -s | grep dev | cut -f 1 -d ' '
/dev/hda3
```

Use the output of the preceding command in this kernel configuration option, and on the kernel boot line where it specifies where the kernel should be resumed from. After the machine has been suspended, to have it resume properly, pass the `resume=/dev/swappartition` argument to the kernel command line to have it use the proper image. If you do not want to have the suspended image restored, use the `noresume` kernel command-line argument.

CPU Frequency Scaling

Most modern processors can slow down the internal clock of the processor to conserve power and battery life. Linux supports this ability and offers a variety of power "governors." Different governors implement different heuristics in order to determine how to vary the processor speed depending on the system load and other variables.

1. Enable the basic frequency scaling functionality:

```
Power management options (ACPI, APM)
    [*] CPU Frequency scaling
```

2. Select the different type of frequency governors you wish to use:

```
Power management options (ACPI, APM)
    [*] CPU Frequency scaling
    [*]   'performance' governor
    [*]   'powersave' governor
    [*]   'userspace' governor for userspace frequency scaling
    [*]   'ondemand' cpufreq policy governor
    [*]   'conservative' cpufreq governor
```

For more information on what the different governors do, see the entry for CPU_FREQ in Chapter 11.

3. Select the default governor you wish to have running when the machine boots:

```
Power management options (ACPI, APM)
    [*] CPU Frequency scaling
          Default CPUFreq governor (performance)
```

4. Select the specific processor type on the machine. For details on how to determine the processor type of the machine, see the earlier section, "Processor Types."

```
Power management options (ACPI, APM)
    [*] CPU Frequency scaling
    ---   CPUFreq processor drivers
    [ ]   ACPI Processor P-States driver
    [ ]   AMD Mobile K6-2/K6-3 PowerNow!
    [ ]   AMD Mobile Athlon/Duron PowerNow!
    [ ]   AMD Opteron/Athlon64 PowerNow!
    [ ]   Cyrix MediaGX/NatSemi Geode Suspend Modulation
    [*]   Intel Enhanced SpeedStep
    [*]     Use ACPI tables to decode valid frequency/voltage pairs
    [*]     Built-in tables for Banias CPUs
    [ ]   Intel Speedstep on ICH-M chipsets (ioport interface)
    [ ]   Intel SpeedStep on 440BX/ZX/MX chipsets (SMI interface)
    [ ]   Intel Pentium 4 clock modulation
    [ ]   nVidia nForce2 FSB changing
    [ ]   Transmeta LongRun
```

Different Memory Models

Linux on 32-bit Intel hardware can access up to 64 GB of memory, but the address space of the 32-bit processor is only 4 GB. To work around this limitation, Linux can map the additional memory into another area and then switch to it when other tasks need it. But if your machine has a smaller amount of memory, it is easier for Linux not to have to worry about handling the bigger areas, so it is beneficial to tell the kernel how much memory you want it to support. For a more detailed description of this option, please see the entry for HIGHMEM in Chapter 11.

Linux supports three different memory models for 32-bit Intel processors, depending on the memory available:

- Under 1 GB of physical memory
- Between 1 and 4 GB of physical memory
- Greater than 4 GB of physical memory

To select the amount of memory:

```
Processor type and features
    High Memory Support
        (X) off
        ( ) 4GB
        ( ) 64GB
```

ACPI

On almost all modern Intel-based systems, ACPI is required in order for the machine to work properly. ACPI is a standard that allows the BIOS of the computer to work with the operating system in order to access the hardware in an indirect manner, in the hope of handling a wide range of devices with relatively little code specific to each operating system. ACPI also provides a facility to help suspend and resume a machine and control the speed of the processor and fans. If you have a laptop, it is recommended that you enable this option.

To enable ACPI:

```
Power management options (ACPI, APM)
    ACPI (Advanced Configuration and Power Interface) Support
        [*] ACPI Support
```

There are a wide range of different ACPI "drivers" that control different types of ACPI devices. You should enable the specific ones that you have on your machine:

```
Power management options (ACPI, APM)
    ACPI (Advanced Configuration and Power Interface) Support
        [*] ACPI Support
        [*]   AC Adapter
        [*]   Battery
        [*]   Button
        [*]   Video
        [*]   Generic Hotkey (EXPERIMENTAL)
        [*]   Fan
        [*]   Processor
        [*]     Thermal Zone
        [ ]   ASUS/Medion Laptop Extras
        [ ]   IBM ThinkPad Laptop Extras
        [ ]   Toshiba Laptop Extras
```

Networking

Networking is required for almost all machines today, and Linux supports almost every networking option available. Here I am going to show only a few of the wide variety that are present.

For all networking options, including different drivers, the main network configuration option must be enabled:

```
Networking
    [*] Networking support
```

The TCP/IP option should also be selected so that the machine can talk to other machines on the Internet:

```
Networking
    [*] Networking support
          Networking options
              [*] TCP/IP networking
```

Netfilter

The Netfilter portion of the Linux kernel is a framework for filtering and manipulating all network packets that pass through the machine. It is commonly used if you wish to enable a firewall on the machine to protect it from different systems on the Internet, or to use the machine as a proxy for other machines on the network. For more details on what Netfilter is good for, please see the entry for NETFILTER in Chapter 11.

1. To enable the main Netfilter option:

```
Networking
    [*] Networking support
          Networking options
              [*] Network packet filtering (replaces ipchains)
```

2. It is recommended that you enable the Netfilter netlink interface and Xtables support when using netlink:

```
Networking
    [*] Networking support
          Networking options
              [*] Network packet filtering (replaces ipchains)
                    Core Netfilter Configuration
                        [*] Netfilter netlink interface
                        [*] Netfilter Xtables support (required for ip_
                            tables)
```

3. The different protocols that you wish to filter should also be selected:

```
Networking
    [*] Networking support
          Networking options
              [*] Network packet filtering (replaces ipchains)
                    IP: Netfilter Configuration
                        [M] Connection tracking (required for masq/NAT)
                        [ ]   Connection tracking flow accounting
                        [ ]   Connection mark tracking support
                        [ ]   Connection tracking events (EXPERIMENTAL)
                        [ ]   SCTP protocol connection tracking support
                              (EXPERIMENTAL)
                        [M]   FTP protocol support
                        [ ]   IRC protocol support
                        [ ]   NetBIOS name service protocol support
                              (EXPERIMENTAL)
                        [M]   TFTP protocol support
                        [ ]   Amanda backup protocol support
                        [ ]   PPTP protocol support
                        [ ]   H.323 protocol support (EXPERIMENTAL)
```

Network Drivers

Linux supports a wide array of different network devices. The most common one is a PCI network device, into which an Ethernet cable can be plugged. To determine whether you have a PCI network device on the system, and what type it is, run the following command:

```
$ /usr/sbin/lspci | grep Ethernet
03:0c.0 Ethernet controller: D-Link System Inc RTL8139 Ethernet (rev 10)
03:0e.0 Ethernet controller: Intel Corporation 82545GM Gigabit Ethernet
Controller (rev 04)
```

Note that your response will probably not be identical; what is important is that the command shows some PCI Ethernet devices.

1. Enable PCI support for the kernel:

   ```
   Bus options (PCI, PCMCIA, EISA, MCA, ISA)
   [*] PCI Support
   ```

2. Enable basic network device support:

   ```
   Device Drivers
       Network device support
           [*] Network device support
   ```

3. Then comes the fun task of finding the specific device drivers for your hardware. The most common place to find Ethernet devices for modern hardware is in the gigabit section of the driver selection:

   ```
   Device Drivers
       Network device support
           [*] Network device support
               Ethernet (1000 Mbit)
   ```

   Some older ethernet devices will be found in the 10- and 100-Mbit section:

   ```
   Device Drivers
       Network device support
           [*] Network device support
               Ethernet (10 or 100Mbit)
   ```

   Look through those sections to find the proper driver for your specific devices.

IrDA

IrDA is an infrared protocol used by a number of laptops and PDAs to communicate over very short distances. It is prevalent on older hardware, with newer hardware using Bluetooth to communicate instead. See the later section, "Bluetooth," for configuring Bluetooth.

1. IrDA is a network protocol, so it can be found under the networking main menu:

   ```
   Networking
       [*] Networking support
       [*]   IrDA (infrared) subsystem support
   ```

2. A number of different IrDA protocols can be selected, depending on the type of device you wish to communicate with and the program used to do the communication:

   ```
   Networking
       [*] Networking support
           --- IrDA (infrared) subsystem support
           ---   IrDA protocols
           [*]   IrLAN protocol (NEW)
   ```

```
        [*]   IrCOMM protocol (NEW)
        [*]   Ultra (connectionless) protocol (NEW)
```

3. There are a wide range of different types of IrDA devices, some serial, some PCI, and others based on USB. To select the specific type of IrDA device you have, choose it under the driver submenu for IrDA:

```
Networking
    [*] Networking support
        --- IrDA (infrared) subsystem support
              Infrared-port device drivers
              --- SIR device drivers
              [ ] IrTTY (uses Linux serial driver)
              --- Dongle support
              --- Old SIR device drivers
              --- Old Serial dongle support
              --- FIR device drivers
              [ ] IrDA USB dongles
              [ ] SigmaTel STIr4200 bridge (EXPERIMENTAL)
              [ ] NSC PC87108/PC87338
              [ ] Winbond W83977AF (IR)
              [ ] Toshiba Type-O IR Port
              [ ] SMSC IrCC (EXPERIMENTAL)
              [ ] ALi M5123 FIR (EXPERIMENTAL)
              [ ] VLSI 82C147 SIR/MIR/FIR (EXPERIMENTAL)
              [ ] VIA VT8231/VT1211 SIR/MIR/FIR
```

Bluetooth

Bluetooth is a wireless technology that was created to replace IrDA to talk between devices over a very short distance. It is a short-range wireless technology that was designed as a replacement for cables, operates within a 10 meter radius, and is commonly used in mobile phones.

1. Bluetooth is a network protocol, so it can be found under the networking main menu:

```
Networking
    [*] Networking support
    [*]   Bluetooth subsystem support
```

2. There are two main protocol selections for Bluetooth. Both of these should be enabled in order to work with all types of Bluetooth devices:

```
Networking
    [*] Networking support
        --- Bluetooth subsystem support
        [*]   L2CAP protocol support
        [*]   SCO links support
```

3. There are relatively few individual Bluetooth devices drivers available, because almost all of these devices follow the Bluetooth specification detailing how devices should operate. The drivers marked in the following list must be selected in order for Bluetooth to work with the device:

```
Networking
    [*] Networking support
        --- Bluetooth subsystem support
```

```
Bluetooth device drivers
    [M] HCI USB driver
    [*]   SCO (voice) support
    [ ] HCI UART driver
    [M] HCI BCM203x USB driver
    [M] HCI BPA10x USB driver
    [ ] HCI BlueFRITZ! USB driver
    [ ] HCI DTL1 (PC Card) driver
    [ ] HCI BT3C (PC Card) driver
    [ ] HCI BlueCard (PC Card) driver
    [ ] HCI UART (PC Card) device driver
    [ ] HCI VHCI (Virtual HCI device) driver
```

Wireless

Wireless networking is very popular, with almost all modern laptops having a built-in wireless network device. Linux supports a wide range of wireless drivers, with more being added every week. To determine whether you have a PCI wireless device on the system, and what type it is, run the following command:

```
$ /usr/sbin/lspci | grep -i wireless
06:05.0 Network controller: Intel Corporation PRO/Wireless 2915ABG MiniPCI
Adapter (rev 05)
```

Note that your response will probably not be identical; what is important is that the command shows some PCI wireless devices.

1. To enable wireless support in Linux, the IEEE 802.11 network configuration option must be enabled. (802.11 is the number of the wireless specification that all these devices follow.)

```
Networking
    [*] Networking support
    [*]   Generic IEEE 802.11 Networking Stack
```

2. Also enable the different 802.11 protocol options and the `Software MAC` option to provide full support for all different types of wireless devices in Linux:

```
Networking
    [*] Networking support
    [*]   Generic IEEE 802.11 Networking Stack
    [*]     IEEE 802.11 WEP encryption (802.1x)
    [M]     IEEE 802.11i CCMP support
    [M]     IEEE 802.11i TKIP encryption
    [M]     Software MAC add-on to the IEEE 802.11 networking stack
```

3. Support for the different PCI types of wireless network devices is found under the Network driver section of the configuration:

```
Device Drivers
    Network device support
        Wireless LAN (non-hamradio)
            [*] Wireless LAN drivers (non-hamradio) & Wireless
            Extensions
            [*]   Wireless Extension API over RtNetlink
```

   There is a wide range of different PCI drivers in this section. Select the proper one depending on the device you have.

The USB wireless networking device drivers are in a different section of the configuration:

```
Device Drivers
    USB Support
        USB Network Adapters
```

Filesystems

Linux supports a wide range of traditional filesystem types and a number of different types of filesystems (volume managers, clustered filesystems, etc.). The traditional filesystem types (normal or journaled) can be selected from the main File systems configuration menu:

```
File systems
    [*] Second extended fs support
    [*] Ext3 journalling file system support
    [ ] Reiserfs support
    [ ] JFS filesystem support
    [ ] XFS filesystem support
```

This section will show a few of the nontraditional filesystem types that Linux supports and how to enable them.

RAID

RAID offers the option of combining numerous disks together so that they look like one logical disk. This can help in providing ways of providing redundancy or speed by spreading the data across different disk platters. Linux supports both hardware and software RAID. Hardware RAID is handled by the disk controller, without any help needed from the kernel.

1. Software RAID is controlled by the kernel, and can be selected as a build option:

```
Device Drivers
    Multi-device support (RAID and LVM)
        [*] Multiple devices driver support (RAID and LVM)
        [*]   RAID support
```

2. There are many different types of RAID configurations. At least one needs to be selected in order for RAID to work properly:

```
Device Drivers
    Multi-device support (RAID and LVM)
        [*] Multiple devices driver support (RAID and LVM)
        [*]   RAID support
        [*]     Linear (append) mode
        [*]     RAID-0 (striping) mode
        [*]     RAID-1 (mirroring) mode
        [*]     RAID-10 (mirrored striping) mode (EXPERIMENTAL)
        [*]     RAID-4/RAID-5 mode
        [*]     RAID-6 mode
```

Logical Volume Manager and Device Mapper

Much like RAID, Logical Volume Manager (LVM) allows the user to combine different block devices to look like one logical device. However, it does not work on a device level like RAID, but through a block and sector mapping mechanism. It allows different portions of different disks to be combined together to look like one large block device to the user. To do this, the kernel uses something called Device Mapper (DM).

1. Enable DM support in the kernel:

```
Device Drivers
    Multi-device support (RAID and LVM)
        [*] Multiple devices driver support (RAID and LVM)
        [*]   Device mapper support
```

2. There are a number of helper modules that work with DM to provide additional functionality. You should enable them if you wish to encrypt your devices, or allow snapshot functionality:

```
Device Drivers
    Multi-device support (RAID and LVM)
        [*] Multiple devices driver support (RAID and LVM)
        [*]   Device mapper support
        [*]     Crypt target support
        [*]     Snapshot target (EXPERIMENTAL)
        [*]     Mirror target (EXPERIMENTAL)
        [*]     Zero target (EXPERIMENTAL)
        [*]     Multipath target (EXPERIMENTAL)
```

File Sharing with Windows

Samba is a program that allows Linux users to access Windows machines natively across the network, providing a way to share drives and devices in a transparent manner. It also allows Linux to work as a Windows server, allowing Windows clients to connect to it thinking that it is a real Windows machine.

Two different filesystems that allow a Linux machine to connect with a Windows machine: the SMB filesystem and the CIFS filesystem. For the ability to connect to older Windows for Workgroups or Windows 95 or 98 machines, select the SMB filesystem:

```
File systems
    Network File Systems
        [*] SMB file system support (to mount Windows shares etc.)
```

For the ability to connect to newer Windows machines, the CIFS filesystem is recommended instead:

```
File systems
    Network File Systems
        [*] CIFS support
```

For more details on the differences between these two filesystems, and when one should be used instead of the other, please see the SMB_FS and CIFS entries in Chapter 11.

OCFS2

OCFS2 is a cluster filesystem from Oracle that works for large network installations and small local systems at the same time. This filesystem is recommended when using large databases, such as Oracle or DB2, because it can be moved over time to different backing disks across the network quite easily as more storage is needed.

To enable the filesystem:

```
File systems
    [*] OCFS2 file system support
```

Security

The Linux kernel supports different security models by providing hooks and letting you build in your choice of model. At the moment, only a few models come with the default kernel source tree, but developers of new models are working on getting more accepted.

Default Linux Capabilities

The standard type of security model for Linux is the "capability" model. You should always select this option unless you really want to run an insecure kernel for some reason.

To enable it:

```
Security options
    [*] Enable different security models
    [*]   Default Linux Capabilities
```

SELinux

A very popular security model is called SELinux. This model is supported by a number of different Linux distributions.

SELinux requires that the networking option be enabled. See the earlier section, "Networking," to enable this.

SELinux also requires that audit be enabled in the kernel configuration. To do this:

```
General setup
    [*] Auditing support
```

Also, the networking security option must be enabled:

```
Security options
    [*] Enable different security models
    [*]   Socket and Networking Security Hooks
```

Now it is possible to select the SELinux option:

```
Security options
    [*] Enable different security models
    [*] NSA SELinux Support
```

There are also a number of individual SELinux options that you might wish to enable. Please see the help for the individual different items for more descriptions on what they do:

```
Security options
    [*] Enable different security models
    [*] NSA SELinux Support
    [ ]   NSA SELinux boot parameter
    [ ]   NSA SELinux runtime disable
    [*]   NSA SELinux Development Support
    [*]   NSA SELinux AVC Statistics
    (1)   NSA SELinux checkreqprot default value
```

Kernel Debugging

A wide range of different kernel options can help in debugging what is going on within the kernel. Following is a list of some of the more common ones that can be useful for discovering new things about how the kernel works, or help find potential problems within the current kernel source code.

Kernel Log Timestamps

The kernel outputs a wide range of messages to its logfile. These messages can be seen by looking at the system logfile (usually located in */var/log/messages*), or by running the *dmesg* command.

Sometimes it is useful to see exactly when those messages were created. *dmesg*, however, does not put any timestamps on the events it shows, and the time resolution of */var/log/messages* is only to the nearest second. You can configure the kernel to assign each message a timestamp that is accurate down to the smallest measurable kernel time value (usually in the microsecond range).

To enable timestamp options on kernel messages:

```
Kernel hacking
    [*] Show timing information on printks
```

Magic SysRq Keys

The SysRq key on the keyboard can be used to control the kernel in a number of different ways while the kernel is running, or after it has crashed.

To enable this option:

```
Kernel hacking
    [*] Magic SysRq key
```

For a full description of the different actions that can be triggered by this option, please see the file *Documentation/sysrq.txt* in the kernel source tree.

Debug Filesystem

A RAM-based filesystem can be used to output a lot of different debugging information. This filesystem is called *debugfs* and can be enabled:

```
Kernel hacking
    [*] Debug filesystem
```

After you enable this option and boot the rebuilt kernel, it creates the directory */sys/kernel/debug* as a location for the user to mount the `debugfs` filesystem. Do this manually by:

```
$ mount -t debugfs none /sys/kernel/debug
```

Or have the filesystem mounted automatically at boot time by adding the following line to the */etc/fstab* file:

```
debugfs /sys/kernel/debug debugfs 0 0
```

After you mount *debugfs*, a large number of different directories and files will turn up in the */sys/kernel/debug/* directory. These are all virtual and dynamically generated by the kernel, like the files in *procfs* or *sysfs*. The files can be used to help debug different kernel subsystems, or just to see what is happening to the system as it runs.

General Kernel Debugging

Here are a range of other good kernel configuration options that you might wish to enable if you want to help kernel developers debug different problems, or just learn more about how the kernel works by looking at the messages that these options print out. Note that if you enable almost any of these options, the kernel will slow down a small amount, so if you notice any decrease in performance, you might wish to disable the options:

```
Kernel hacking
    [*] Kernel debugging
    [*]   Detect Soft Lockups
    [ ]   Collect scheduler statistics
    [*]   Debug slab memory allocations
    [*]     Memory leak debugging
    [*]   Mutex debugging, deadlock detection
    [*]   Spinlock debugging
    [*]   Sleep-inside-spinlock checking
    [ ]   kobject debugging
    [ ]   Highmem debugging
    [ ]   Compile the kernel with debug info
```