



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*.



[\*] 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:



## 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.



```
[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
```





- ☐ K6/K6-III/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
  - ☒ 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



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
```



## 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:



- [\*] 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
            [ ] SSMC 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
```





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
```



## 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:



## 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 *procs* 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
```