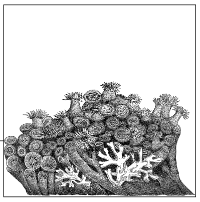
11



Kernel Configuration Option Reference

This chapter lists the most important configuration options offered when you run *make config* or one of its graphical interfaces. The majority of the chapter is based on the in-kernel documentation for the different kernel configuration options, which were written by the kernel developers and released under the GPL.

EXPERIMENTAL Prompt for development and/or incomplete code/drivers

Some of the many things that Linux supports (such as network drivers, filesystems, network protocols, etc.) can be in a state of development where the functionality, stability, or the level of testing is not yet high enough for general use. This is usually known as the "alpha-test" phase among developers. If a feature is currently in alpha-test, the developers usually discourage uninformed widespread use of this feature by the general public to avoid "Why doesn't this work?" mail messages. However, active testing and use of these systems is welcomed. Just be aware that it may not meet the normal level of reliability or may fail to work in some special cases. Detailed bug reports from people familiar with the kernel internals are usually welcomed by the developers. (But before submitting bug reports, please read the documents README, MAINTAINERS, REPORTING-BUGS, *Documentation/BUG-HUNTING*, and *Documentation/oops-tracing.txt* in the kernel source.)

This option also makes obsolete drivers available. These are drivers that have been replaced by something else and/or are scheduled to be removed in a future kernel release.

Unless you intend to help test and develop a feature or driver that falls into this category, or you have a situation that requires using these features, you should probably say `no` here, which will cause the configurator to present you with fewer choices. If you say yes here, you will be offered the choice of using features or drivers that are currently considered to be in the alpha-test phase.

On its own, this option does not do anything except allow you to select other options.

LOCALVERSION

Local version—append to kernel release

This allows you to append an extra string to the end of your kernel version. This will show up when you enter a *uname* command, for example. The string you set here will be appended after the contents of any files with a filename beginning with *localversion* in your object and source tree, in that order. The string can be a maximum of 64 characters.

AUDIT

Auditing support

Enable an auditing infrastructure that can be used with another kernel subsystem, such as SELinux (which requires this for logging of avc messages output).

IKCONFIG

Kernel *.config* support

This option enables the complete Linux kernel *.config* file contents to be saved in the kernel. It documents which kernel options are used in a running kernel or an on-disk kernel. This information can be extracted from the kernel image file with the script *scripts/extract-ikconfig* and used as input to rebuild the current kernel or to build another kernel. It can also be extracted from a running kernel by reading the file */proc/config.gz*.

EMBEDDED

Configure standard kernel features (for small systems)

This option allows certain base kernel options and settings to be disabled or tweaked. This is for specialized environments that can tolerate a “nonstandard” kernel. This is recommend only for experts, as it is very easy to change the options to create a kernel that will not even boot properly.

On its own, this option does not do anything except allow you to select other options.

MODULES

Enable loadable module support

Kernel modules are small pieces of compiled code that can be inserted in the running kernel, rather than being permanently built into the kernel. If you select this option, many parts of the kernel can be built as modules (by answering `M` instead of `yes` where indicated): this is most useful for infrequently used options that are not required for booting. For more information, see Chapter 4 and the manpages for *modprobe*, *lsmod*, *modinfo*, *insmod*, and *rmmod*.

If you say yes here, you will need to run *make modules_install* to put the modules under */lib/modules* where the module tools can find them.

IOSCHED_NOOP

No-op I/O scheduler

The no-op I/O scheduler is a minimal scheduler that does basic merging and sorting. Its main uses include nondisk-based block devices such as memory devices and specialized software or hardware environments that do their own scheduling and require only minimal assistance from the kernel.

IOSCHED_AS

Anticipatory I/O scheduler

The anticipatory I/O scheduler is the default disk scheduler. It is generally a good choice for most environments, but is quite large and complex compared to the deadline I/O scheduler. It can also be slower in some cases, especially under some database loads.

IOSCHED_DEADLINE

Deadline I/O scheduler

The deadline I/O scheduler is simple and compact. It is often as good as the anticipatory I/O scheduler, and under some database workloads, even better. In the case of a single process performing I/O to a disk at any one time, its behavior is almost identical to the anticipatory I/O scheduler and so is a good choice.

IOSCHED_CFQ

CFQ I/O scheduler

The CFQ I/O scheduler tries to distribute bandwidth equally among all processes in the system. It should provide a fair working environment, suitable for desktop systems.

SMP

Symmetric multiprocessing support

This enables support for systems with more than one CPU. If you have a system with only one CPU, like most personal computers, say no. If you have a system with more than one CPU, say yes.

If you say no here, the kernel will run on single and multiprocessor machines, but will use only one CPU of a multiprocessor machine. If you say yes here, the kernel will run on many, but not all, single-processor machines. On a single-processor machine, the kernel will run faster if you say no here.

Note that if you say yes here and choose architecture 586 or Pentium under Processor family, the kernel will not work on 486 architectures. Similarly, multiprocessor kernels for the PPro architecture may not work on all Pentium-based boards.

See also *Documentation/smp.txt*, *Documentation/i386/IO-APIC.txt*, *Documentation/nmi_watchdog.txt*, and the SMP-HOWTO available at *http://www.tldp.org/docs.html#howto*.

M386 `386`

This is the processor type of your CPU. This information is used for optimization. In order to compile a kernel that can run on all x86 CPU types (albeit not optimally fast), you can specify 386 here.

The kernel will not necessarily run on earlier architectures than the one you have chosen; e.g., a Pentium-optimized kernel will run on a PPro, but not necessarily on a i486.

Here are the settings recommended for greatest speed:

`386`
: Choose this if you have an AMD/Cyrix/Intel 386DX/DXL/SL/SLC/SX, Cyrix/TI 486DLC/DLC2, UMC 486SX-S, or NexGen Nx586 processor. Only 386 kernels will run on a 386 class machine.

`486`
: Choose this if you have an AMD/Cyrix/IBM/Intel 486DX/DX2/DX4, SL/SLC/SLC2/SLC3/SX/SX2 and UMC U5D, or U5S processor.

`586`
: Choose this if you have a generic Pentium processor lacking the TSC (timestamp counter) register.

`Pentium-Classic`
: Choose this if you have an Intel Pentium processor.

`Pentium-MMX`
: Choose this if you have an Intel Pentium MMX processor.

`Pentium-Pro`
: Choose this if you have an Intel Pentium Pro processor.

`Pentium-II`
: Choose this if you have an Intel Pentium II or pre-Coppermine Celeron processor.

`Pentium-III`
: Choose this if you have an Intel Pentium III or Coppermine Celeron processor.

`Pentium-4`
: Choose this if you have an Intel Pentium 4 or P4-based Celeron processor.

`K6`
: Choose this if you have an AMD K6, K6-II or K6-III (aka K6-3D) processor.

`Athlon`
: Choose this if you have an AMD K7 family (Athlon/Duron/Thunderbird) processor.

`Crusoe`
: Choose this if you have a Transmeta Crusoe series processor.

`Efficeon`
: Choose this if you have a Transmeta Efficeon series processor.

`Winchip-C6`
: Choose this if you have an original IDT Winchip processor.

`Winchip-2`
: Choose this if you have an IDT Winchip 2 processor.

`Winchip-2`
: Choose this if you have an IDT Winchip processor with 3DNow! capabilities.

`GeodeGX1`
: Choose this if you have a Geode GX1 (Cyrix MediaGX) processor.

`Geode GX/LX`
: Choose this if you have an AMD Geode GX or LX processor.

`CyrixIII/VIA C3`
: Choose this if you have a VIA Cyrix III or VIA C3 processor.

`VIA C3-2`
: Choose this if you have a VIA C3-2 "Nehemiah" (model 9 and above) processor.

If you don't know what to do, choose 386.

X86_GENERIC

Generic x86 support

Instead of just including optimizations for the selected x86 variant (e.g., PII, Crusoe, or Athlon), include some more generic optimizations as well. This will make the kernel perform better on x86 CPUs other than the one selected.

This is really intended for distributors who need more generic optimizations.

NR_CPUS

Maximum number of CPUs (2-255)

This allows you to specify the maximum number of CPUs that this kernel will support. The maximum supported value is 255 and the minimum value that makes sense is 2.

This option is purely to save memory; each supported CPU adds approximately 8 KB to the kernel image.

SCHED_SMT

SMT (HyperThreading) scheduler support

SMT scheduler support improves the CPU scheduler's decision-making on Intel Pentium 4 chips with HyperThreading, at a cost of slightly increased overhead in some places.

PREEMPT_NONE No forced preemption (server)

This is the traditional Linux preemption model, geared toward maximizing throughput. It still provides good latency most of the time, occasional longer delays are possible.

Select this option if you are building a kernel for a server or scientific/computation system, or if you want to maximize the raw processing power of the kernel, irrespective of scheduling latencies.

PREEMPT_ VOLUNTARY Voluntary kernel preemption (desktop)

This option reduces the latency of the kernel by adding more “explicit preemption points” to the kernel code. These new preemption points have been selected to reduce the maximum latency of rescheduling, which provides faster response to applications at the cost of slighly lower throughput.

This option speeds up reaction to interactive events by allowing a low-priority process to voluntarily preempt itself even if it is in kernel mode executing a system call. This allows applications to appear to run more smoothly even when the system is under load.

Select this if you are building a kernel for a desktop system.

PREEMPT Preemptible kernel (low-latency desktop)

This option reduces the latency of the kernel by making all kernel code (except code executing in a critical section) preemptible. This allows reaction to interactive events by permitting a low priority process to be preempted involuntarily even if the processor is in kernel mode executing a system call and would otherwise not be about to reach a natural preemption point. This allows applications to appear to run more smoothly even when the system is under load, at the cost of slighly lower throughput and a slight runtime overhead to kernel code.

Select this if you are building a kernel for a desktop or an embedded system with latency requirements in the milliseconds range.

PREEMPT_BKL Preempt the Big Kernel Lock

This option reduces the latency of the kernel by making the Big Kernel Lock preemptible.

Say yes here if you are building a kernel for a desktop system.

NOHIGHMEM High memory configuration

Linux can use up to 64 GB of physical memory on x86 systems. However, the address space of 32-bit x86 processors is only 4 GB in size. That means that, if you have a large amount of physical memory, not all of it can be permanently mapped by the kernel. The physical memory that's not permanently mapped is called *high memory*.

If you are compiling a kernel that will never run on a machine with more than 1 GB total physical RAM, answer `off` here (the default choice, and suitable for most users). This will result in a 3 GB/1 GB split: 3 GB are mapped so that each process sees a 3 GB virtual memory space and the remaining part of the 4 GB virtual memory space is used by the kernel to permanently map as much physical memory as possible.

If the machine has between 1 and 4 GB physical RAM, answer `4GB` here.

If more than 4 GB is used, answer `64GB` here. This selection turns Intel PAE (Physical Address Extension) mode on. PAE implements three-level paging on IA32 processors. PAE is fully supported by Linux, and PAE mode is implemented on all recent Intel processors (Pentium Pro and better).

If you say `64GB` here, the kernel will not boot on CPUs that don't support PAE!

The actual amount of total physical memory will either be autodetected or can be forced by using a kernel command line option such as `mem=256M`. (See Chapter 9 for details about how to pass options to the kernel at boot time, and what options are available.)

If unsure, say `off`.

HIGHMEM4G 4GB

Select this if you have a 32-bit processor and between 1 and 4 GB of physical RAM.

HIGHMEM64G 64GB

Select this if you have a 32-bit processor and more than 4 GB of physical RAM.

FLATMEM_ MANUAL Flat memory

This option allows you to change some of the ways that Linux manages its memory internally. Most users will see only have one option here: `FLATMEM`. This is normal and a correct option.

Some users of more advanced features, such as `NUMA` and memory hotplug, may have different options here. `DISCONTIGMEM` is a more mature, better tested system, but is incompatible with memory hotplug and may suffer decreased performance over `SPARSEMEM`. If you are unsure between sparse memory and discontiguous memory, choose discontiguous memory.

If unsure, choose this option, flat memory.

DISCONTIGMEM_MANUAL

Discontiguous memory

This option provides better support than flat memory for discontiguous memory systems. These systems have holes in their physical address spaces, and this option handles the holes more efficiently. However, the vast majority of hardware has quite flat address spaces and can experience degraded performance from the extra overhead this option imposes.

Many NUMA configurations will have this as the only option.

If unsure, choose flat memory over this option.

SPARSEMEM_MANUAL

Sparse memory

This will be the only option for some systems, including memory hotplug systems.

For many other systems, this will be an alternative to discontiguous memory. This option provides some potential performance benefits, along with decreased code complexity, but it is newer and more experimental.

If you are unsure, choose discontiguous memory or flat memory.

SECCOMP

Enable seccomp to safely compute untrusted bytecode

This kernel feature is useful for number-crunching applications that may need to compute untrusted bytecode during their execution. By using pipes or other transports made available to the process as file descriptors supporting the read/write syscalls, it's possible to isolate those applications in their own address space using seccomp. Once seccomp is enabled via */proc/pid/seccomp*, it cannot be disabled and the task is allowed to execute only a few safe syscalls defined by each seccomp mode.

If you are unsure, say yes. Only embedded systems should be built by answering no.

KEXEC

kexec system call (experimental)

kexec is a system call that implements the ability to shut down your current kernel and start up another. It is like a reboot, but is independent of the system firmware. And like a reboot, you can start any kernel with it, not just Linux.

The name comes from the similarity to the exec system call.

Do not be surprised if this code does not initially work for you. It may help to enable device hotplugging support. As of this writing, the exact hardware interface is strongly in flux, so no good recommendation can be made.

HOTPLUG_CPU Support for hot-pluggable CPUs (experimental)

Say yes here to experiment with turning CPUs off and on, and to enable suspend on SMP systems. CPUs can be controlled through the */sys/devices/system/cpu* interface.

PM Power management support

Power management allows parts of your computer to shut off or be put into a power-conserving sleep mode if they are not being used. There are two competing standards for doing this: APM and ACPI. If you want to use either one, say yes here and then also enable one of those two standards.

Power management is most important for battery-powered laptop computers; if you have a laptop, check out the Linux Laptop home page at *http://www.linux-on-laptops.com*, Tuxmobil-Linux on Mobile Computers at *http://www.tuxmobil.org*, and the "Battery Powered Linux" mini-HOWTO at *http://www.tldp.org/docs.html#howto*.

Note that, even if you say `no` here, Linux on the x86 architecture will issue the HLT instruction if nothing is being done, thereby sending the processor to sleep and saving power.

SOFTWARE_SUSPEND Software suspend

Enable machine suspension.

When the machine is suspended, an image is saved in your active swap. Upon next boot, pass the `resume=`*`/dev/swappartition`* argument to the kernel to have it detect the saved image, restore memory state from it, and continue to run as before. If you do not want the previous state to be reloaded, use the `noresume` kernel argument. However, note that your partitions will be *fsck*'d and you must issue *mkswap* on your swap partitions again. The procedure does not work with swap files.

Right now you may boot without resuming and then resume later, but in the meantime you cannot use those swap partitions/files that were involved in suspending. In this case, also, there is a risk that buffers on disk won't match with saved ones.

For more information, see *Documentation/power/swsusp.txt*.

ACPI ACPI Support

Advanced Configuration and Power Interface (ACPI) support for Linux requires ACPI-compliant hardware and firmware, and assumes the presence of OS-directed configuration and power management (OSPM) software. This option will enlarge your kernel by about 70 KB.

Linux ACPI provides a robust functional replacement for several legacy configuration and power management interfaces, including the Plug and Play BIOS specification (PnP BIOS), the MultiProcessor specification (MPS), and the Advanced Power Management (APM) specification. If both ACPI and APM support are configured, whichever is loaded first will be used.

The ACPI SourceForge project at *http://sourceforge.net/projects/acpi* contains the latest source code, documentation, tools, mailing list subscription, and other information.

Linux support for ACPI is based on Intel Corporation's ACPI Component Architecture (ACPI CA). For more information, see *http://developer.intel.com/technology/iapc/acpi*.

ACPI is an open industry specification codeveloped by Compaq, Intel, Microsoft, Phoenix, and Toshiba. The specification is available at *http://www.acpi.info*.

CPU_FREQ

CPU frequency scaling

CPU frequency scaling allows you to change the clock speed of CPUs on the fly. This can save power, because the lower the CPU clock speed, the less power the CPU consumes.

Note that this driver doesn't automatically change the CPU clock speed; you need to either enable a dynamic CPUFreq policy governor (described later) after booting or use a userspace tool.

For details, take a look at *Documentation/cpu-freq*.

CPU_FREQ_DEFAULT_GOV_PERFORMANCE

Performance

Use the CPUFreq performance governor. This sets the frequency statically to the highest frequency supported by the CPU.

CPU_FREQ_DEFAULT_GOV_USERSPACE

Userspace

Use the CPUFreq userspace governor. This allows you to set the CPU frequency manually and allows a userspace program to set the CPU dynamically without requiring you to first enable the userspace governor manually.

CPU_FREQ_GOV_PERFORMANCE

"Performance" CPUFreq policy governor

This CPUFreq policy governor sets the frequency statically to the highest available CPU frequency.

CPU_FREQ_GOV_POWERSAVE	"Powersave" CPUFreq policy governor This sets the frequency statically to the lowest available CPU frequency.
CPU_FREQ_GOV_USERSPACE	"Userspace" CPUFreq policy governor Enable this CPUFreq policy governor either when you want to set the CPU frequency manually or when a userspace program should be able to set the CPU dynamically, as on LART (*http://www.lart-maker.nl*). For details, take a look at *Documentation/cpu-freq*.
CPU_FREQ_GOV_ONDEMAND	"Ondemand" CPUFreq policy governor This driver adds a dynamic CPUFreq policy governor. The governor polls the CPU and changes its frequency based on CPU utilization. Support for this governor depends on the CPU's ability to do fast frequency switching (i.e., very low latency frequency transitions). For details, take a look at *Documentation/cpu-freq*.
CPU_FREQ_GOV_CONSERVATIVE	"Conservative" CPUFreq policy governor This driver is similar to the Ondemand governor both in its source code and its purpose. The difference is that the Conservative governor is optimized for a battery-powered system. The frequency is gracefully increased and decreased rather than jumping to 100 percent when speed is required. If you are using a laptop, a PDA, or an AMD64-based computer (due to the unacceptable step-by-step latency issues between the minimum and maximum frequency transitions in the CPU), you will probably want to use this governor. If you have a desktop machine, consider the Ondemand governor instead. For details, take a look at *Documentation/cpu-freq*.
PCI	PCI support PCI is a bus system used by the processor to talk to internal devices and add-on cards. It is extremely common and found in almost all modern computers. Say yes to this option unless you have a special reason not to.
PCCARD	PCCard (PCMCIA/CardBus) support Say yes here if you want to attach PCMCIA or PC cards to your Linux computer. These are credit-card size devices such as network cards, modems, or hard drives often used with laptop computers.

There are actually two varieties of these cards: 16-bit PCMCIA and 32-bit CardBus cards.

PCMCIA 16-bit PCMCIA support

This option enables support for 16-bit PCMCIA cards. Most older PC cards are 16-bit PCMCIA cards, so unless you know you're only using 32-bit CardBus cards, say yes here.

To use 16-bit PCMCIA cards, you will need supporting software in most cases. See the file *Documentation/Changes* for location and details.

CARDBUS 32-bit CardBus support

CardBus is a bus mastering architecture for PC cards, which allows for 32-bit PC cards (the original PCMCIA standard specifies only a 16-bit wide bus). Many newer PC cards are actually CardBus cards.

To use 32-bit PC cards, you also need a CardBus-compatible host bridge. Virtually all modern PCMCIA bridges do this, and most of them are "yenta-compatible," so enable that option too.

HOTPLUG_PCI Support for PCI hotplug (experimental)

Say yes here if you have a motherboard with a PCI hotplug controller. This allows you to add and remove PCI cards while the machine is powered up and running.

NET Networking support

Say yes here unless you are an expert with a really good reason not to. The reason is that some programs need kernel networking support even when running on a standalone machine that isn't connected to any other computer.

If you are upgrading from an older kernel, you should consider updating your networking tools too, because changes in the kernel and the tools often go hand in hand. The tools are contained in the *net-tools* package, the location and version number of which are given in *Documentation/Changes*.

For a general introduction to Linux networking, it is highly recommended that you read the NET-HOWTO, available from *http://www.tldp.org/docs.html#howto*.

UNIX Unix domain sockets

If you say yes here, you will include support for Unix domain sockets; sockets are the standard Unix mechanism for establishing and accessing network connections. Many commonly used programs such as the X Window System, *syslog*, and *udev* use these sockets even if your machine is not connected to any network. Unless you are working on an embedded system or something similar, you definitely want to say yes here.

INET TCP/IP networking

These are the protocols used on the Internet and on most local Ethernets. It is highly recommended that you say yes here, since some programs (e.g., the X Window System) use TCP/IP even if your machine is not connected to any other computer. They use the so-called *loopback device*, which this option sets up. It will enlarge your kernel by about 144 KB.

For an excellent introduction to Linux networking, please read the "Linux Networking" HOWTO, available from *http://www.tldp.org/docs.html#howto*.

IP_ADVANCED_ROUTER IP: advanced router

If you intend to run your Linux box mostly as a router, i.e., as a computer that forwards and redistributes network packets, say yes here. You will then be presented with several options that allow more precise control about the routing process.

The answer to this question won't directly affect the kernel: answering no will just cause the configurator to skip all the questions about advanced routing.

Note that your box can act as a router only if you enable IP forwarding in your kernel; you can do that by saying yes to the */proc* filesystem support and Sysctl support options and executing the line:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

at boot time after the */proc* filesystem has been mounted.

If you turn on IP forwarding, you will also get rp_filter, which automatically rejects incoming packets if the routing table entry for their source address doesn't match the network interface they're arriving on. This has security advantages because it prevents IP spoofing; however, it can pose problems if you use asymmetric routing (packets from you to a host take a different path from packets that go from that host to you) or if you operate a nonrouting host that has several IP addresses on different interfaces. To turn rp_filter off, enter:

```
echo 0 > /proc/sys/net/ipv4/conf/device/rp_filter
```

or:

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
```

NETFILTER Network packet filtering

Netfilter is a framework for filtering and mangling network packets that pass through your Linux box.

The most common use of packet filtering is to run your Linux box as a firewall protecting a local network from the Internet. The type of firewall provided by this kernel support is called a *packet filter*, which means that it can reject individual network packets based on type, source, destination, etc. The other kind of firewall, a *proxy-based* one, is more secure but more intrusive and more bothersome to set up; it inspects the network traffic much more closely, modifies it, and has knowledge about the higher-level protocols, which a packet filter lacks. Moreover, proxy-based firewalls often require changes to the programs running on the local clients. Proxy-based firewalls don't need support by the kernel, but they are often combined with a packet filter, which works only if you say yes here.

You should also say yes here if you intend to use your Linux box as the gateway to the Internet for a local network of machines without globally valid IP addresses. This is called *masquerading*. If one of the computers on your local network wants to send something to the outside, your box can "masquerade" as that computer, i.e., it forwards the traffic to the intended outside destination, but modifies the packets to make it look like they came from the firewall box itself. Masquerading works both ways: if the outside host replies, the Linux box will silently forward the traffic to the correct local computer. This way, the computers on your local net are completely invisible to the outside world, even though they can reach the outside and can receive replies. It is even possible to run globally visible servers from within a masqueraded local network using a mechanism called port forwarding. Masquerading is also often called NAT (Network Address Translation). Other operating systems often call this term PAT (Port Address Translation).

Another use of Netfilter is in transparent proxying: if a machine on the local network tries to connect to an outside host, your Linux box can transparently forward the traffic to a local server, typically a caching proxy server.

Yet another use of Netfilter is building a bridging firewall. Using a bridge with Network packet filtering enabled makes iptables "see" the bridged traffic. For filtering on the lower network and Ethernet protocols over the bridge, use ebtables (located under bridge Netfilter configuration).

Various modules exist for Netfilter that replace the previous masquerading (*ipmasqadm*), packet-filtering (*ipchains*), transparent proxying, and port-forwarding mechanisms. Please see *Documentation/Changes* under *iptables* for the location of these packages.

Chances are that you should say yes here if you compile a kernel which will run as a router and no for regular hosts.

NET_SCHED QoS and/or fair queueing

When the kernel has several packets to send out over a network device, it has to decide which ones to send first, which ones to delay, and which ones to drop. This is the job of queueing disciplines. Several different algorithms for how to do this “fairly” have been proposed.

If you say `no` here, you will get the standard packet scheduler, which is a FIFO (first come, first served) scheduler. If you say yes here, you will be able to choose from among several alternative algorithms that can then be attached to different network devices. This is useful, for example, if some of your network devices are real-time devices that need a certain minimum data flow rate, or if you need to limit the maximum data flow rate for traffic that matches specified criteria.

To administer these schedulers, you’ll need the user-level utilities from the package *iproute2+tc* at *http://linux-net.osdl.org/index.php/Iproute2*.

This Quality of Service (QoS) support will enable you to use Differentiated Services (diffserv) and Resource Reservation Protocol (RSVP) on your Linux router if you also say yes to the corresponding options. Documentation and software is at *http://diffserv.sourceforge.net*.

IRDA IrDA (infrared) subsystem support

Say yes here if you want to build support for the IrDA protocols. The Infrared Data Association specifies standards for wireless infrared communication and is supported by most laptops and PDAs.

To use Linux support for the IrDA protocols, you will also need some userspace utilities such as *irattach*. For more information, see the file *Documentation/networking/irda.txt*. You also want to read the IR-HOWTO, available at *http://www.tldp.org/docs.html#howto*.

If you want to exchange bits of data (e.g., vCal, vCard) with a PDA, you will need to install an OBEX application, such as OpenObex from *http://sourceforge.net/projects/openobex*.

IRLAN IrLAN protocol

Say yes here if you want to build support for the IrLAN protocol. IrLAN emulates an Ethernet and makes it possible to put up a wireless LAN using infrared beams.

The IrLAN protocol can be used to talk with infrared access points such as the HP NetbeamIR or the ESI JetEye NET. You can also connect to another Linux machine running the IrLAN protocol for ad hoc networking.

IRNET

IrNET protocol

Say yes here if you want to build support for the IrNET protocol. IrNET is a PPP driver, so you will also need a working PPP subsystem (driver, daemon, and configuration).

IrNET is an alternate way to transfer TCP/IP traffic over IrDA. It uses synchronous PPP over a set of point to point IrDA sockets. You can use it between Linux machines or with Windows.

IRCOMM

IrCOMM protocol

Say yes here if you want to build support for the IrCOMM protocol. IrCOMM implements serial port emulation, and makes it possible to use all existing applications that understand ttys with infrared links. Thus, you should be able to use applications such as PPP and *minicom*.

IRDA_ULTRA

Ultra (connectionless) protocol

Say yes here to support the connectionless Ultra IRDA protocol. Ultra allows you to exchange data over IrDA with really simple devices (watch, beacon) without the overhead of the IrDA protocol (no handshaking, no management frames, simple fixed header). Ultra is available as a special socket: `socket(AF_IRDA, SOCK_DGRAM, 1)`.

BT

Bluetooth subsystem support

Bluetooth is a low-cost, low-power, and short-range wireless technology. It was designed as a replacement for cables and other short-range technologies such as IrDA. Bluetooth operates in a personal area range that typically extends up to 10 meters. More information about Bluetooth can be found at *http://www.bluetooth.com*.

The Linux Bluetooth subsystem consist of several layers:

Bluetooth core
: HCI device and connection manager, scheduler

HCI device drivers
: Interface to the hardware

SCO module
: SCO audio links

L2CAP module
: Logical Link Control and Adaptation Protocol

RFCOMM module
: RFCOMM Protocol

BNEP
: Module Bluetooth Network Encapsulation Protocol

CMTP
: Module CAPI Message Transport Protocol

HIDP
Module Human Interface Device Protocol

To use the Linux Bluetooth subsystem, you will need several user-space utilities, such as *hciconfig* and *hcid*. These utilities and updates to Bluetooth kernel modules are provided in the BlueZ packages at *http://www.bluez.org*.

IEEE80211 Generic IEEE 802.11 networking stack

This option enables the hardware-independent IEEE 802.11 networking stack.

MTD Memory Technology Device (MTD) support

Memory Technology Devices are flash, RAM, and similar chips, often used for solid-state filesystems on embedded devices. This option provides the generic support for MTD drivers to register themselves with the kernel and for potential users of MTD devices to enumerate the devices present and obtain a handle on them. It also allows you to select individual drivers for particular hardware and users of MTD devices.

PARPORT Parallel port support

If you want to use devices connected to your machine's parallel port (the connector at the computer with 25 holes), e.g., a printer, ZIP drive, or Parallel Line Internet Protocol (PLIP) link, you need to say yes here.

Please read *Documentation/parport.txt* and *drivers/parport/BUGS-parport* for more information. For extensive information about drivers for many devices attaching to the parallel port, see *http://www.torque.net/linux-pp.html*.

It is possible to share a single parallel port among several devices, and it is safe to compile all the corresponding drivers into the kernel. If you have more than one parallel port and want to specify which port and IRQ will be used by this driver at module load time, take a look at *Documentation/parport.txt*.

PNP Plug and Play support

Plug and Play (PnP) is a standard for peripherals that allows them to be configured by software—for example, to assign IRQs or other parameters. No jumpers on the cards are needed; instead, the values are provided to the cards from the BIOS, from the operating system, or using a userspace utility.

Say yes here if you would like Linux to configure your PnP devices. You should then also say yes to all of the protocols needed. Alternatively, you can say no here and configure your PnP devices using userspace utilities such as the *isapnptools* package.

ISAPNP

ISA Plug and Play support

Say yes here if you would like support for ISA PnP devices. Some information is available in *Documentation/isapnp.txt*.

If you use have ISA Plug and Play devices, please use the ISA PnP tools found at *http://www.roestock.demon.co.uk/isapnptools* to configure them properly.

PNPBIOS

Plug and Play BIOS support (experimental)

Linux uses the PNPBIOS defined in "Plug and Play BIOS Specification Version 1.0A May 5, 1994" to autodetect built-in mainboard resources (e.g., parallel port resources).

If you would like the kernel to detect and allocate resources to your mainboard devices (on some systems they are disabled by the BIOS) say yes here. The PNPBIOS can also help prevent resource conflicts between mainboard devices and other bus devices.

ACPI is expected to supersede PNPBIOS some day. Currently, they coexist nicely. If you have a non-ISA system that supports ACPI, you probably don't need PNPBIOS support.

IDE

ATA/ATAPI/MFM/RLL support

If you say yes here, your kernel will be able to manage low-cost mass storage units such as ATA/(E)IDE and ATAPI. The most common examples of such devices are IDE hard drives and ATAPI CD-ROM drives.

If your system is pure SCSI and doesn't use these interfaces, you can say no here.

- Integrated Disk Electronics (IDE, also known as ATA-1) is a connecting standard for mass storage units such as hard disks. It was designed by Western Digital and Compaq Computer in 1984. It was then named ST506. Several disks use the IDE interface.
- AT Attachment (ATA) is the superset of the IDE specifications. ST506 is also called ATA-1.
- Fast-IDE is ATA-2 (also named Fast ATA).
- Enhanced IDE (EIDE) is ATA-3. It provides support for larger disks (up to 8.4 GB by means of the LBA standard), more disks (four instead of two), and for other mass storage units, such as tapes and CD-ROMs.
- UDMA/33 (also known as UltraDMA/33) is ATA-4. By using fast DMA controllers, it provides faster transfer modes (with less load on the CPU) than previous PIO (Programmed processor Input/Output) from previous ATA/IDE standards.
- ATA Packet Interface (ATAPI) is a protocol used by EIDE tape and CD-ROM drives, similar in many respects to the SCSI protocol.

SMART IDE (self-monitoring, -analysis, and -reporting technology) was designed in order to prevent data corruption and disk crashes by detecting pre-hardware failure conditions (heat, access time, and the like). Disks built after June 1995 may follow this standard. The kernel itself doesn't manage this; however, there are quite a number of user programs, such as *smart*, that can query the status of SMART parameters from disk drives.

For further information, please read *Documentation/ide.txt*.

BLK_DEV_IDE Enhanced IDE/MFM/RLL disk/CD-ROM/tape/floppy support

If you say yes here, you will use the full-featured IDE driver to control up to 10 ATA/IDE interfaces, each one able to serve a "master" and a "slave" device, for a total of up to 20 ATA/IDE disk/CD-ROM/tape/floppy drives.

Useful information about large (540 MB) IDE disks, multiple interfaces, what to do if ATA/IDE devices are not automatically detected, sound card ATA/IDE ports, module support, and other topics is contained in *Documentation/ide.txt*. For detailed information about hard drives, consult the Disk-HOWTO and the Multi-Disk-HOWTO, available at *http://www.tldp.org/docs.html#howto*.

To fine-tune ATA/IDE drive/interface parameters for improved performance, look for the *hdparm* package at *ftp://ibiblio.org/pub/Linux/system/hardware*.

Do not compile this driver as a module if your root filesystem (the one containing the directory /) is located on an IDE device.

If you have one or more IDE drives, enable this option. If your system has no IDE drives or if memory requirements are really tight, you could say `no` here, and select the old hard disk driver option instead to save about 13 KB of memory in the kernel.

BLK_DEV_IDEDISK Include IDE/ATA-2 disk support

This includes enhanced support for MFM/RLL/IDE hard disks. If you have a MFM/RLL/IDE disk and there is no special reason to use the old hard disk driver instead, say `yes`. If you have an SCSI-only system, you can say `no` here.

Do not compile this driver as a module if your root filesystem (the one containing the directory /) is located on the IDE disk.

BLK_DEV_IDECD Include IDE/ATAPI CD-ROM support

If you have a CD-ROM drive using the ATAPI protocol, say yes here. ATAPI is a newer protocol used by IDE CD-ROM and tape drives, similar to the SCSI protocol. Most new CD-ROM drives use ATAPI, including the NEC-260, Mitsumi FX400, Sony 55E, and just about all non-SCSI double (2×) or better speed drives.

If you say yes here, the CD-ROM drive will be identified at boot time along with other IDE devices, as something such as `hdb` or `hdc` (check the boot messages using the *dmesg* command). If this is your only CD-ROM drive, you can say `no` to all other CD-ROM options, but be sure to also enable the ISO 9660 CD-ROM filesystem support option.

Note that older versions of LILO (LInux LOader) cannot properly deal with IDE/ATAPI CD-ROMs, so install LILO 16 or higher, available from *http://lilo.go.dyndns.org*.

BLK_DEV_IDEFLOPPY

Include IDE/ATAPI floppy support

If you have an IDE floppy drive that uses the ATAPI protocol, answer yes. ATAPI is a newer protocol used by IDE CD-ROM/tape/floppy drives, similar to the SCSI protocol.

The LS-120 and the IDE/ATAPI Iomega ZIP drive are also supported by this driver. For information about jumper settings and the question of when a ZIP drive uses a partition table, see *http://www.win.tue.nl/~aeb/linux/zip/zip-1.html*. (ATAPI PD-CD/CDR drives are not supported by this driver; support for PD-CD/CDR drives is available if you answer yes to SCSI emulation support).

If you say yes here, the floppy drive will be identified along with other IDE devices, with a name such as `hdb` or `hdc` (check the boot messages using the *dmesg* command).

SCSI

SCSI device support

If you want to use a SCSI hard disk, SCSI tape drive, SCSI CD-ROM, or any other SCSI device under Linux, say yes and make sure that you know the name of your SCSI host adapter (the card inside your computer that "speaks" the SCSI protocol, also called SCSI controller), because you will be asked for it.

You also need to say yes here if you have a device that speaks the SCSI protocol. Examples of these include the parallel port version of the IOMEGA ZIP drive, USB storage devices, Fibre Channel, FireWire storage, and the IDE-SCSI emulation driver.

Do not compile this as a module if your root filesystem (the one containing the directory /) is located on a SCSI device.

BLK_DEV_SD

SCSI disk support

If you want to use SCSI hard disks, Fibre Channel disks, USB storage, or the SCSI or parallel port version of the IOMEGA ZIP drive, say yes and read the SCSI-HOWTO, the Disk-HOWTO, and the Multi-Disk-HOWTO, available from *http://www.tldp.org/docs.html#howto*. This is *not* for SCSI CD-ROMs.

Do not compile this driver as a module if your root filesystem (the one containing the directory /) is located on a SCSI disk. In this case, do not compile the driver for your SCSI host adapter as a module either.

CHR_DEV_ST SCSI tape support

If you want to use a SCSI tape drive under Linux, say yes and read the SCSI-HOWTO, available from *http://www.tldp.org/docs.html#howto*, and *Documentation/scsi/st.txt* in the kernel source. This is *not* for SCSI CD-ROMs.

BLK_DEV_SR SCSI CD-ROM support

If you want to use a SCSI or FireWire CD-ROM under Linux, say yes and read the SCSI-HOWTO and the CDROM-HOWTO at *http://www.tldp.org/docs.html#howto* for more directions. Also make sure to enable the ISO 9660 CD-ROM filesystem support option.

CHR_DEV_SG SCSI generic support

If you want to use SCSI scanners, synthesizers, or CD writers, or just about anything having "SCSI" in its name other than hard disks, CD-ROMs, or tapes, say yes here. These won't be supported by the kernel directly, so you need some additional software that knows how to talk to these devices using the SCSI protocol.

For scanners, look at SANE *http://www.sane-project.org*. For CD writer software look at Cdrtools, *http://cdrecord.berlios.de/old/private/cdrecord.html*, and for burning a "disk at once," check out CDRDAO, *http://cdrdao.sourceforge.net*. Cdparanoia is a high-quality digital reader of audio CDs (*http://www.xiph.org/paranoia*). For other devices, it's possible that you'll have to write the driver software yourself. Please read the file *Documentation/scsi/scsi-generic.txt* for more information.

CHR_DEV_SCH SCSI media changer support

This is a driver for SCSI media changers. The most common such devices are tape libraries and MOD/CD-ROM jukeboxes. This option is for real jukeboxes; you don't need it for tiny six-slot CD-ROM changers. Media changers are listed as "Type: Medium Changer" in */proc/scsi/scsi*. Check *Documentation/scsi/scsi-changer.txt* for details.

SCSI_MULTI_LUN Probe all LUNs on each SCSI device

If you have a SCSI device, such as a CD jukebox, that supports more than one LUN (Logical Unit Number), and only one LUN is detected, you can say yes here to force the SCSI driver to probe for multiple LUNs. A SCSI device with multiple LUNs acts logically like multiple SCSI devices. The vast majority of SCSI devices have only one LUN, and so most people can say `no` here. The `max_luns` boot/module parameter allows you to override this setting.

SCSI_SATA

Serial ATA (SATA) support

This driver family supports serial ATA host controllers and devices.

MD

Multiple devices driver support (RAID and LVM)

This option supports multiple physical spindles through a single logical device and is required for RAID and logical volume management.

BLK_DEV_MD

RAID support

This driver lets you combine several hard disk partitions into one logical block device. This can be used to simply append one partition to another one or to combine several redundant hard disks into a RAID 1, RAID 4, or RAID 5 device to provide protection against hard disk failures. This is called *software RAID* because the combining of the partitions is done by the kernel. *Hardware RAID* means that the combining is done by a dedicated controller. If you have such a controller, you do not need to say yes here.

More information about software RAID on Linux is in the "Software RAID" mini-HOWTO, available from *http://www.tldp.org/docs.html#howto*. There you will also learn where to get the supporting userspace *raidtools* utilities.

BLK_DEV_DM

Device mapper support

Device mapper is a low-level volume manager. It works by allowing people to specify mappings for ranges of logical sectors. Various mapping types are available, in addition to which people may write their own modules containing custom mappings.

Higher-level volume managers such as LVM2 use this driver.

IEEE1394

IEEE 1394 (FireWire) support

IEEE 1394 describes a high-performance serial bus, which is also known as FireWire or i.Link and is used for connecting all sorts of devices (most notably, digital video cameras) to your computer.

If you have FireWire hardware and want to use it, say yes here. This is the core support only. You will also need to select a driver for your IEEE 1394 adapter.

,ch11.12684 Page 144 Friday, December 1, 2006 10:07 AM

I2O I2O support

The Intelligent Input/Output (I2O) architecture allows hardware drivers to be split into two parts: an operating-system-specific module called the OSM and a hardware-specific module called the HDM. The OSM can talk to a whole range of HDMs, and ideally the HDMs are not OS-dependent. This allows for the same HDM driver to be used under different operating systems if the relevant OSM is in place. In order for this to work, you need to have an I2O interface adapter card in your computer. This card contains a special I/O processor (IOP), allowing high speeds because the CPU does not have to deal with I/O.

If you say yes here, you will get a choice of interface adapter drivers and OSMs and will have to enable the correct ones.

NETDEVICES Network device support

You can say no here if you do not intend to connect your Linux box to any other computer.

You'll have to say yes if your computer contains a network card that you want to use under Linux. If you are going to run SLIP or PPP over a telephone line or null modem cable you also need to say yes here. Connecting two machines with parallel ports using PLIP needs this, as well as AX.25/KISS, for sending Internet traffic over amateur radio links.

See also the *Linux Network Administrator's Guide* by Tony Bautts et al. (O'Reilly), available at *http://www.tldp.org/guides.html*.

NET_ETHERNET Ethernet (10 or 100 Mbit)

Ethernet (also called IEEE 802.3 or ISO 8802-2) is the most common type of Local Area Network (LAN) in universities and companies.

Common varieties of Ethernet are 10-base2 or Thinnet (10 Mbps over coaxial cable, linking computers in a chain), 10-baseT or twisted pair (10 Mbps over twisted pair cable, linking computers to central hubs), 10-baseF (10 Mbps over optical fiber links, using hubs), 100-baseTX (100 Mbps over two twisted pair cables, using hubs), 100-baseT4 (100 Mbps over four standard voice-grade twisted pair cables, using hubs), 100-baseFX (100 Mbps over optical fiber links), and gigabit Ethernet (1 Gbps over optical fiber or short copper links). The 100-base varieties are also known as Fast Ethernet.

If your Linux machine will be connected to an Ethernet and you have an Ethernet network interface card (NIC) installed in your computer, say yes here and read the Ethernet-HOWTO, available from *http://www.tldp.org/docs.html#howto*. You will then also have to say yes to the driver for your particular NIC.

Note that the answer to this question won't directly affect the kernel: saying `no` will just cause the configurator to skip all the questions about Ethernet network cards.

NET_RADIO

Wireless LAN drivers (non-hamradio) and Wireless Extensions

Support for wireless LANs and everything having to do with packet radio, but not with amateur radio or FM broadcasting.

Saying yes here also enables the Wireless Extensions, creating */proc/net/wireless* and enabling *iwconfig* access. The Wireless Extensions are a generic API that allows a driver to expose configuration and statistics for common wireless LANs to userspace. Wireless Extensions provide a single set of tools that can support all the variations of wireless LANs, regardless of their type (as long as the driver supports Wireless Extensions). Another advantage is that these parameters may be changed on the fly without restarting the driver or operating system. If you wish to use Wireless Extensions with wireless PCMCIA cards (PC cards), you need to say yes here. You can fetch the tools from *http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html*.

PPP

PPP (Point-to-Point Protocol) support

PPP sends Internet traffic over telephone (and other serial) lines. Ask your access provider if they support it, because otherwise you can't use it. An older protocol with the same purpose is called SLIP. Most Internet access providers these days support PPP rather than SLIP.

To use PPP, you need an additional program called *pppd* as described in the PPP-HOWTO, available at *http://www.tldp.org/docs.html#howto*. Make sure that you have the version of *pppd* recommended in *Documentation/Changes*. The PPP option enlarges your kernel by about 16 KB.

There are actually two versions of PPP: the traditional PPP for asynchronous lines, such as regular analog phone lines, and synchronous PPP, which can be used over digital ISDN lines, for example. If you want to use PPP over phone lines or other asynchronous serial lines, you need to enable the PPP support for async serial ports option.

PPPOE

PPP over Ethernet (experimental)

Support for PPP over Ethernet.

This driver requires the latest version of *pppd* from the CVS repository at *cvs.samba.org*. Alternatively, see the *RoaringPenguin* package *http://www.roaringpenguin.com/pppoe*, which contains instruction on how to use this driver under the heading "Kernel mode PPPoE."

ISDN

ISDN support

ISDN (Integrated Services Digital Networks, called RNIS in France) is a special type of fully digital telephone service; it's mostly used to connect to your Internet service provider (with SLIP or PPP). The main advantage of ISDN is that the speed is higher than ordinary modem/telephone connections and that you can have voice conversations while downloading stuff. It works only if your computer is equipped with an ISDN card and both you and your service provider purchased an ISDN line from the phone company. For details, read *http://www.alumni.caltech.edu/~dank/isdn*.

Select this option if you want your kernel to support ISDN.

PHONE

Linux telephony support

Say yes here if you have a telephony card, which, for example, allows you to use a regular phone for voice over IP applications.

This option has nothing to do with modems. You do not need to say yes here in order to be able to use a modem under Linux.

INPUT

Generic input layer (needed for keyboard, mouse, ...)

Say yes here if you have any input device (mouse, keyboard, tablet, joystick, steering wheel, etc.) connected to your system and want it to be available to applications. This includes a standard PS/2 keyboard and mouse.

Say `no` here if you have a headless system (no monitor or keyboard).

More information is available in *Documentation/input/input.txt*.

VT

Virtual terminal

Say yes here to get support for terminal devices with display and keyboard devices. These are called "virtual" because you can run several virtual terminals (also called virtual consoles) on one physical terminal.

You need at least one virtual terminal device in order to make use of your keyboard and monitor. Therefore, only people configuring an embedded system would want to say `no` here in order to save some memory. The only way to log into such a system is then via a serial or network connection.

Virtual terminals are useful because, for example, one virtual terminal can display system messages and warnings, another one can be used for a text-mode user session, and a third could run an

X session, all in parallel. Switching between virtual terminals is done with certain key combinations, usually Alt-function key.

If you are unsure, say `yes`, or else you won't be able to do much with your Linux system.

VT_CONSOLE Support for console on virtual terminal

The system console is the device that receives all kernel messages and warnings and allows logins in single user mode. If you answer yes here, a virtual terminal (the device used to interact with a physical terminal) can be used as system console. This is the most common mode of operations, so you should say `yes` unless you want the kernel messages be output only to a serial port (in which case you should also enable the console on 8250/16550 and compatible serial port option).

If you say `yes` here, the currently visible virtual terminal (*/dev/tty0*) will be used as system console by default. You can change that with a kernel command-line option such as `console=tty3`, which specified the third virtual terminal as the system console. (See Chapter 9 for details about how to pass options to the kernel at boot time, and what options are available.)

SERIAL_8250 8250/16550 and compatible serial support

This selects whether you want to include the driver for the standard serial ports. The standard answer is `yes`. People who might say `no` here are those setting up dedicated Ethernet WWW/FTP servers, or a user that has one of the various bus mice instead of a serial mouse and doesn't intend to use his machine's standard serial port for anything. In addition, the Cyclades and Stallion multiserial port drivers do not need this driver.

Do not compile this driver as a module if you are using nonstandard serial ports, because the configuration information will be lost when the driver is unloaded. This limitation may be lifted in the future.

Most people will say yes here, so that they can use serial mice, modems, and similar devices connected to the standard serial ports.

AGP */dev/agpgart* (AGP Support)

AGP (Accelerated Graphics Port) is a bus system used mainly to connect graphics cards to the rest of the system.

If you have an AGP system and you say yes here, it will be possible to use the AGP features of your 3D rendering video card. This code acts as a sort of "AGP driver" for the motherboard's chipset.

If you need more texture memory than you can get with the AGP GART (theoretically up to 256 MB, but in practice usually 64 or 128 MB due to kernel allocation issues), you could use PCI accesses and have up to a couple of gigabytes of texture space.

Note that this is the only way to have X and GLX use write-combining with MTRR support on the AGP bus. Without this option, OpenGL direct rendering will be a lot slower, but still faster than PIO.

You should say yes here if you want to use GLX or DRI.

DRM

Direct Rendering Manager (XFree86 4.1.0 and higher DRI support)

Kernel-level support for the Direct Rendering Infrastructure (DRI) was introduced in XFree86 4.0. If you say yes here, you need to select the module that's right for your graphics card from the list. These modules provide support for synchronization, security, and DMA transfers. Please see *http://dri.sourceforge.net* for details. You should also select and configure AGP (*/dev/agpgart*) support.

I2C

I2C support

I2C (pronounced "I-square-C") is a slow serial bus protocol developed by Philips and used in many micro controller applications. SMBus, or System Management Bus, is a subset of the I2C protocol. More information is contained in the directory *Documentation/i2c*, especially in the file there called *summary*.

Both I2C and SMBus are supported by this option. You will need it for hardware sensors support and Video For Linux support.

If you want I2C support, in addition to saying yes here, you must also select the specific drivers for your bus adapters.

SPI

SPI support

The Serial Peripheral Interface (SPI) is a low-level synchronous protocol. Chips that support SPI can have data transfer rates up to several tens of Mbps. Chips are addressed with a controller and a chipselect. Most SPI slaves don't support dynamic device discovery; some are even write-only or read-only.

SPI is widely used by microcontrollers to talk with sensors, EEPROM and flash memory, codecs and various other controller chips, analog-to-digital and digital-to-analog converters, and more. MMC and SD cards can be accessed using SPI protocol, and for DataFlash cards used in MMC sockets, SPI must always be used.

SPI is one of a family of similar protocols using a four-wire interface (select, clock, data in, and data out), including Microwire (half duplex), SSP, SSI, and PSP. This driver framework should work with most such devices and controllers.

HWMON Hardware-monitoring support

Hardware-monitoring devices let you monitor the hardware health of a system. Most modern motherboards include such a device. It may include temperature sensors, voltage sensors, fan speed sensors, and various additional features such as the ability to control the speed of the fans. If you want this support you should say yes here and also to the specific driver for your sensor chip.

VIDEO_DEV Video for Linux

This option enables support for audio/video capture and overlay devices and FM radio cards. The exact capabilities of each device vary.

The kernel includes support for the new Video for Linux Two API, (V4L2) as well as the original system. Drivers and applications need to be rewritten to use V4L2, but drivers for popular cards and applications for most video capture functions already exist.

Additional info and docs are available at *http://linuxtv.org*. Documentation for V4L2 is also available at *http://bytesex.org/v4l*.

DVB DVB for Linux

This option enables support for Digital Video Broadcasting hardware. Enable this if you own a DVB adapter and want to use it or if you are compiling Linux for a digital set-top box.

API specs and user tools are available from *http://www.linuxtv.org*.

FB Support for frame buffer devices

The frame buffer device provides an abstraction for the graphics hardware. It represents the frame buffer of some video hardware and allows application software to access the graphics hardware through a well-defined interface, so the software doesn't need to know anything about the low-level (hardware register) stuff.

Frame buffer devices work identically across the different architectures supported by Linux and make the implementation of application programs easier and more portable. At this point, an X server exists that uses the frame buffer device exclusively. On several non-X86 architectures, the frame buffer device is the only way to use the graphics hardware.

You need a program called *fbset* to make full use of frame buffer devices. Please read *Documentation/fb/framebuffer.txt* and the Framebuffer-HOWTO, available at *http://www.tldp.org/HOWTO/Framebuffer-HOWTO.html* for more information.

Say yes here and to the driver for your graphics board if you are compiling a kernel for a non-x86 architecture. If you are compiling for the x86 architecture, you can say yes if you want to use the frame buffer, but it is not essential.

Please note that running graphical applications that directly touch the hardware (e.g., an accelerated X server) and that are not attuned to the frame buffer device may cause unexpected results.

VGA_CONSOLE VGA text console

Saying yes here will allow you to use Linux in text mode through a display that complies with the generic VGA standard. Virtually everyone wants that.

The program SVGATextMode can be used to utilize SVGA video cards to their full potential in text mode. Download it from *ftp://ibiblio.org/pub/Linux/utils/console*.

LOGO Bootup logo

This option enables the pretty penguin logo at boot time. It will show up on the frame buffer while the kernel is booting. The number of penguins shows the number of processors that the kernel has found.

SOUND Sound card support

If you have a sound card in your computer—i.e., if it can create more than an isolated beep—say yes. Be sure to have all the information about your sound card and its configuration (I/O port, interrupt and DMA channel), because you will be asked for it.

Read the Sound-HOWTO, available from *http://www.tldp.org/docs.html#howto*. General information about the modular sound system is contained in the file *Documentation/sound/oss/Introduction*. The file *Documentation/sound/oss/README.OSS* contains some slightly outdated but still useful information as well. Newer sound driver documentation can be found in files in the *Documentation/sound/alsa* directory.

If you have a PnP sound card and you want to configure it at boot time using the ISA PnP tools (read *http://www.roestock.demon.co.uk/isapnptools*), you need to compile sound card support as a module and load that module after the PnP configuration is finished. To do this properly, read *Documentation/sound/oss/README.modules*.

I'm told that even without a sound card, you can make your computer create more than an occasional beep by programming the PC speaker. Kernel patches and supporting utilities to do that are in the *pcsp* package, available at *ftp://ftp.infradead.org/pub/pcsp*.

SND

Advanced Linux Sound Architecture

Say yes to enable ALSA (Advanced Linux Sound Architecture), the standard Linux sound system.

For more information, see *http://www.alsa-project.org*.

SND_USB_AUDIO

USB Audio/MIDI driver

Say yes here to include support for USB audio and USB MIDI devices.

USB

Support for host-side USB

Universal Serial Bus (USB) is a specification for a serial bus subsystem that offers higher speeds and more features than the traditional PC serial port. The bus supplies power to peripherals and allows for hot swapping. Up to 127 USB peripherals can be connected to a single USB host in a tree structure.

The USB host is the root of the tree, the peripherals are the leaves, and the inner nodes are special USB devices called hubs. Most PCs now have USB host ports, used to connect peripherals such as scanners, keyboards, mice, modems, cameras, disks, flash memory, network links, and printers to the PC.

Say yes here if your computer has a host-side USB port and you want to use USB devices. You then need to say yes to at least one of the Host Controller Driver (HCD) options that follow. Choose a USB 1.1 controller, such as UHCI HCD support or OHCI HCD support, and EHCI HCD (USB 2.0) support except for older systems that do not have USB 2.0 support. It does not hurt to select them all if you are not certain.

If your system has a device-side USB port, used in the peripheral side of the USB protocol, see the USB Gadget option instead.

After choosing your HCD, select drivers for the USB peripherals you'll be using. You may want to check out the information provided in *Documentation/usb* and especially the links given in *Documentation/usb/usb-help.txt*.

USB_EHCI_HCD

EHCI HCD (USB 2.0) support

The Enhanced Host Controller Interface (EHCI) is standard for USB 2.0 "high-speed" (480 Mbit/sec, 60 Mbyte/sec) host controller hardware. If your USB host controller supports USB 2.0, you will likely want to configure this HCD. At the time of this writing, the primary implementation of EHCI is a chip from NEC, widely available in add-on PCI cards, but implementations are in the works from other vendors, including Intel and Philips. Motherboard support is emerging.

EHCI controllers are packaged with “companion” host controllers (OHCI or UHCI) to handle USB 1.1 devices connected to root hub ports. Ports will connect to EHCI if the device is high-speed; otherwise, they connect to a companion controller. If you configure EHCI, you should probably configure the OHCI (for NEC and some other vendors) USB HCD or UHCI (for VIA motherboards) HCD, too.

You may want to read *Documentation/usb/ehci.txt* for more information on this driver.

USB_OHCI_HCD OHCI HCD support

The Open Host Controller Interface (OHCI) is a standard for accessing USB 1.1 host controller hardware. It does more in hardware than Intel’s UHCI specification. If your USB host controller follows the OHCI spec, say yes. On most non-x86 systems, and on x86 hardware that’s not using a USB controller from Intel or VIA, this is appropriate. If your host controller doesn’t use PCI, this is probably appropriate. For a PCI-based system where you’re not sure, the *lspci -v* command will list the right prog-if for your USB controller(s): EHCI, OHCI, or UHCI.

USB_UHCI_HCD UHCI HCD (most Intel and VIA) support

The Universal Host Controller Interface is a standard created by Intel for accessing the USB hardware in the PC (which is also called the USB host controller). If your USB host controller conforms to this standard, you may want to say yes. All recent boards with Intel PCI chipsets (such as Intel 430TX, 440FX, 440LX, 440BX, i810, i820) conform to this standard. All VIA PCI chipsets (like VIA VP2, VP3, MVP3, Apollo Pro, Apollo Pro II, or Apollo Pro 133) also use the standard.

USB_STORAGE USB mass storage support

Say yes here if you want to connect USB mass storage devices to your computer’s USB port. This is the driver you need for USB floppy drives, USB hard disks, USB tape drives, USB CD-ROMs, USB flash devices, and memory sticks, along with similar devices. This driver may also be used for some cameras and card readers.

This option enables the SCSI option, but you probably also need SCSI device support: SCSI disk support for most USB storage devices to work properly.

USB_SERIAL USB serial converter support

Say yes here if you have a USB device that provides normal serial ports, or acts like a serial device, and you want to connect it to your USB bus.

Please read *Documentation/usb/usb-serial.txt* for more information on the specifics of the different devices that are supported and on how to use them.

USB_GADGET

Support for USB gadgets

USB is a master/slave protocol, organized with one master host (such as a PC) controlling up to 127 peripheral devices. The USB hardware is asymmetric, which makes it easier to set up: you can't connect a "to-the-host" connector to a peripheral.

Linux can run in the host or in the peripheral. In both cases you need a low-level bus controller driver and some software that talks to it. Peripheral controllers can be either discrete silicon or integrated with the CPU in a microcontroller. The more familiar host-side controllers have names like such as EHCI, OHCI, or UHCI, and are usually integrated into southbridges on PC motherboards.

Enable this configuration option if you want to run Linux inside a USB peripheral device. Configure one hardware driver for your peripheral/device side bus controller, and a "gadget driver" for your peripheral protocol. (If you use modular gadget drivers, you may configure more than one.)

If in doubt, say `no` and don't enable these drivers; most people don't have this kind of hardware (except maybe inside Linux PDAs).

For more information, see *http://www.linux-usb.org/gadget* and the kernel DocBook documentation for this API.

MMC

MMC support

MMC is the MultiMediaCard bus protocol.

If you want MMC support, you should say yes here and also to the specific driver for your MMC interface.

INFINIBAND

InfiniBand support

Core support for InfiniBand. Make sure to also select any protocols you wish to use as well as drivers for your InfiniBand hardware.

EDAC

EDAC core system error reporting (experimental)

EDAC is designed to report errors in the core system. These are low-level errors that are reported by the CPU or supporting chipset: memory errors, cache errors, PCI errors, thermal throttling, etc.

If this code is reporting problems on your system, please see the EDAC project web pages for more information: *http://bluesmoke.sourceforge.net* and *http://buttersideup.com/edacwiki*.

EXT2_FS Second extended filesystem support

ext2 is a standard Linux filesystem for hard disks. Most systems use the upgrade, *ext3*, instead.

Note that the filesystem of your root partition (the one containing the directory /) cannot be compiled as a module without using a special boot process, so building it as a module could be dangerous.

EXT3_FS Third extended filesystem support

This is the journaling version (called *ext3*) of the second extended filesystem, the de facto standard Linux filesystem for hard disks.

The journaling code included in this driver means you do not have to run *fsck* (filesystem checker) on your filesystems after a crash. The journal keeps track of any changes that were being made at the time the system crashed, and can ensure that your filesystem is consistent without the need for a lengthy check.

Other than adding the journal to the filesystem, the on-disk format of *ext3* is identical to *ext2*. It is possible to freely switch between using the *ext3* driver and the *ext2* driver, as long as the filesystem has been cleanly unmounted, or *fsck* is run on the filesystem before the switch.

To add a journal on an existing *ext2* filesystem or change the behavior of *ext3* filesystems, you can use the *tune2fs* utility. To modify attributes of files and directories on *ext3* filesystems, use *chattr*. You need *e2fsprogs* version 1.20 or later in order to create *ext3* journals (available at *http://sourceforge.net/projects/e2fsprogs*).

REISERFS_FS ReiserFS support

This is a journaled filesystem that stores not just filenames but the files themselves in a balanced tree. Balanced trees can be more efficient than traditional filesystem architectural foundations.

In general, ReiserFS is as fast as *ext2*, but is more efficient with large directories and small files.

JFS_FS JFS filesystem support

This is a port of IBM's Journaled Filesystem (JFS). More information is available in the file *Documentation/filesystems/jfs.txt*.

XFS_FS XFS filesystem support

XFS is a high-performance journaling filesystem that originated on the SGI IRIX platform. It is completely multithreaded; supports large files and large filesystems, extended attributes, and variable block sizes; is extent-based; makes extensive use of B-trees; and uses directories, extents, and free space to aid both performance and scalability.

Refer to the documentation at *http://oss.sgi.com/projects/xfs* for complete details. This implementation is on-disk compatible with the IRIX version of XFS.

OCFS2_FS

OCFS2 filesystem support (experimental)

OCFS2 is a general-purpose, extent-based, shared-disk cluster filesystem with many similarities to *ext3*. It supports 64-bit inode numbers and has automatically extending metadata groups, which may also make it attractive for nonclustered use.

You'll want to install the *ocfs2-tools* package in order to at least get the *mount.ocfs2* program.

The project web page is *http://oss.oracle.com/projects/ocfs2* and the tools web page is *http://oss.oracle.com/projects/ocfs2-tools*. OCFS2 mailing lists can be found at *http://oss.oracle.com/projects/ocfs2/mailman*.

INOTIFY

inotify file change notification support

Say yes here to enable inotify support and the associated system calls. inotify is a file change notification system and a replacement for dnotify. inotify fixes numerous shortcomings in dnotify and introduces several new features. It allows monitoring of both files and directories via a single open `fd` object. Other features include multiple file events, one-shot support, and unmount notification.

For more information, see *Documentation/filesystems/inotify.txt*.

QUOTA

Quota support

If you say yes here, you will be able to set per-user limits for disk usage (also called disk quotas). Currently, it works for the *ext2*, *ext3*, and ReiserFS filesystem. *ext3* also supports journaled quotas, for which you don't need to run *quotacheck* after an unclean shutdown. For further details, read the "Quota" mini-HOWTO, available from *http://www.tldp.org/docs.html#howto* or the documentation provided with the quota tools. Quota support is probably useful only for multiuser systems.

AUTOFS_FS

Kernel automounter support

The automounter is a tool that automatically mounts remote filesystems on demand. This implementation is partially kernel-based to reduce overhead when a system is already mounted. This is unlike the BSD automounter (*amd*), which is a pure userspace daemon.

To use the automounter, you need the userspace tools from the autofs package; you can find the location in *Documentation/Changes*. You also want to answer yes to the NFS filesystem support option.

If you want to use the newer version of the automounter with more features, say `no` here and say yes to the Kernel automounter v4 support option.

If you are not a part of a fairly large, distributed network, you probably do not need an automounter, and can say `no` here.

FUSE_FS Filesystem in userspace support

With FUSE it is possible to implement a fully functional filesystem in a userspace program.

There's also companion library named *libfuse*. This library, along with utilities, is available from the FUSE homepage: *http://fuse.sourceforge.net*.

See *Documentation/filesystems/fuse.txt* for more information. See *Documentation/Changes for library/utility* version you need.

If you want to develop a userspace filesystem, or if you want to use a filesystem based on FUSE, answer yes here.

SMB_FS SMB filesystem support (to mount Windows shares etc.)

SMB (Server Message Block) is the protocol Windows for Workgroups (WfW), Windows 95/98, Windows NT and later variants, and OS/2 LAN Manager use to share files and printers over local networks. Saying yes here allows you to mount their filesystems (often called "shares" in this context) and access them just like any other Unix directory. Currently, this works only if the Windows machines use TCP/IP as the underlying transport protocol, not NetBEUI. For details, read *Documentation/filesystems/smbfs.txt* and the SMB-HOWTO, available from *http://www.tldp.org/docs.html#howto*.

If you just want your box to act as an SMB server and make files and printing services available to Windows clients (which need to have a TCP/IP stack), you don't need to say yes here; you can use the Samba set of daemons and programs (available from *ftp://ftp.samba.org/pub/samba*).

CIFS CIFS support (advanced network filesystem for Samba, Window, and other CIFS compliant servers)

This is the client VFS module for the Common Internet File System (CIFS) protocol, which is the successor to the Server Message Block (SMB) protocol, the native file-sharing mechanism for most early PC operating systems. The CIFS protocol is fully supported by file servers such as Windows 2000 (including Windows 2003, NT 4, and Windows XP) as well by Samba (which provides excellent CIFS server support for Linux and many other operating systems). Limited support for Windows ME and similar servers is provided as well. You must use the *smbfs* client filesystem to access older SMB servers such as OS/2 and DOS.

The intent of the *cifs* module is to provide an advanced network filesystem client for mounting local filesystems to CIFS-compliant servers, including support for DFS (hierarchical namespace), secure per-user session establishment, safe distributed caching (*oplock*), optional packet signing, Unicode and other internationalization improvements, and optional Winbind (*nsswitch*) integration. You do not need to enable *cifs* if you are running only a server (Samba). It is possible to enable both *smbfs* and *cifs* (e.g., if you are using CIFS for accessing Windows 2003 and Samba 3 servers, and *smbfs* for accessing old servers). If you need to mount to Samba or Windows from this machine, say yes to this option.

PROFILING Profiling support (experimental)

Say yes here to enable the extended profiling support mechanisms used by profilers such as OProfile.

OPROFILE OProfilesystem profiling (experimental)

OProfile is a profiling system capable of profiling the whole system, including the kernel, kernel modules, libraries, and applications.

For more information and links to the userspace tools needed to use OProfile properly, see the main project page at *http://oprofile.sourceforge.net/news*.

KPROBES Kprobes (experimental)

Kprobes allows you to trap the CPU at almost any kernel address and execute a callback function. `register_kprobe()` establishes a probepoint and specifies the callback. Kprobes is useful for kernel debugging, nonintrusive instrumentation, and testing.

PRINTK_TIME Show timing information on printks

Selecting this option causes timing information to be included in `printk` (kernel message) output. This allows you to measure the interval between kernel operations, including bootup operations. This is useful for identifying long delays in kernel startup.

MAGIC_SYSRQ Magic SysRq key

If you say yes here, you will have some control over the system even if the system crashes for example during kernel debugging (i.e., you will be able to flush the buffer cache to disk, reboot the system immediately, or dump some status information). This is accomplished by pressing various keys while holding down the SysRq (Alt+PrintScreen) key. It also works on a serial console (on PC hardware at least), if you send a BREAK and then within 5 seconds a command keypress. The keys are documented in *Documentation/sysrq.txt*. Don't say yes unless you really know what this hack does.

DEBUG_KERNEL Kernel debugging

Say yes here if you are developing drivers or trying to debug and identify kernel problems.

On its own, this option does not do anything except allow you to chance to select other options.

DEBUG_FS Debug filesystem

debugfs is a virtual filesystem where kernel developers put debugging files. Enable this option to be able to read and write to these files.

SECURITY Enable different security models

This allows you to configure different security modules into your kernel.

If this option is not selected, the default Linux security model will be used.

SECURITY_ SELINUX NSA SELinux support

This selects NSA Security-Enhanced Linux (SELinux). You will also need a policy configuration and a labeled filesystem. You can obtain the policy compiler (*checkpolicy*), the utility for labeling filesystems (*setfiles*), and an example policy configuration from *http://www.nsa.gov/selinux*.