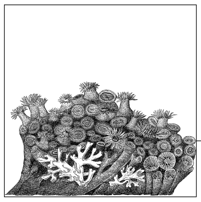

9

Kernel Boot Command-Line Parameter Reference

The majority of this chapter is based on the in-kernel documentation for the different kernel boot command-line reference options, which were written by the kernel developers and released under the GPL.

There are three ways to pass options to the kernel and thus control its behavior:

- When building the kernel. Most of this book discusses these options.
- When starting the kernel. Usually, parameters are passed to the kernel when it is invoked from a boot file such as the GRUB or LILO configuration file.
- At runtime, by writing to files in the */proc* and */sys* directories.

This chapter describes the second method of passing options. The chapter breaks the boot time options into different logical sections. A number of architecture-specific and individual driver options are not listed here. For a complete list of all known options, please see the file *Documentation/kernel-parameters.txt* in the kernel source tree and the individual architecture-specific documentation files.

Not all of the listed options are always available. Most are associated with subsystems and work only if the kernel is configured with those subsystems built in. They also depend on the presence of the hardware with which they are associated.

All of these parameters are case-sensitive.

Module-Specific Options

In addition to the options listed in this chapter, parameters for modules that are built in to the kernel can also be passed on the command line. (Dynamically loaded modules, of course, are not in memory when the kernel boots and therefore cannot be passed as parameters at boot time.) The syntax for passing parameters consists of the module name followed by a dot (`.`) and the parameter.

For example, the *usbcore* module accepts the parameter *blinkenlights* to display flashing lights on all supported USB 2.0 hubs (don't ever say the kernel developers don't have a sense of humor). To set this parameter when loading the module dynamically, you would enter:

```
$ modprobe usbcore blinkenlights=1
```

But if the *usbcore* module is built into the kernel, you achieve the same effect by invoking the kernel with the following option:

```
usbcore.blinkenlights=1
```

Most module options for modules that are built into the kernel can also be changed at runtime by writing to files in the subdirectory named after the module under the */sys/module* directory. Thus, the *blinkenlights* option is represented by the file */sys/module/usbcore/blinkenlights*.

Console Options

These options deal with the console or kernel log, where kernel debugging and error information are displayed.

console

Output console device and options.

`console=Options`

`tty`*n*
: Use the virtual console device *n*.

`ttyS`*n*`[,`*options*`]`, `ttyUSB0[,`*options*`]`
: Use the specified serial port. The options are of the form *bbbbpnf*, where *bbbb* is the baud rate, *p* is parity (`n`, `o`, or `e`), *n* is number of bits, and *f* is flow control (`r` for RTS or omitted). Default is `9600n8`.

: See the file *Documentation/serial-console.txt* for more information on how to use a serial console. If you wish to have access to the kernel console information and do not have a serial port, see the *netconsole* command-line option.

`uart,io,`*addr*`[,`*options*`]`, `uart,mmio,`*addr*`[,`*options*`]`
: Start an early, polled-mode console on the 8250/16550 UART at the specified I/O port or MMIO address, switching to the specified ttyS device later. The options are the same as for ttyS shown earlier.

netconsole

Output console data across the network.

`netconsole=[`*src-port*`]@[`*src-ip*`]/[`*dev*`],[`*target-port*`]@`*target-ip*`/[`*target-mac-address*`]`

Send kernel console data across the network using UDP packets to another machine. Options are:

src-port
: Source port for the UDP packets. The default value is 6665.

src-ip
: Source IP address of the interface to use.

dev
: Network interface to use. `eth0` is an example. The network interface can also run normal network traffic, because the netconsole data is not intrusive and should cause no slowdown in other network operations.

target-port
: Port that the logging agent will use. The default value is 6666.

target-ip
: IP address for the logging agent.

target-mac-address
: Ethernet MAC address for the logging agent.

To listen to this data, the remote machine can use the *syslogd* program, or run the *netcat* program as follows:

```
netcat -u -l -p port
```

For more background on how to use this option, see the file *Documentation/networking/netconsole.txt*.

debug

Enable kernel debugging.

Cause the kernel log level to be set to the debug level, so that all debug messages will be printed to the console at boot time.

quiet

Disable all log messages.

Set the default kernel log level to KERN_WARNING (4), which suppresses all messages during boot except extremely serious ones. (Log levels are defined under the *loglevel* parameter.)

earlyprintk

Show early boot messages.

```
earlyprintk=[vga|serial][,ttySn[,baudrate]][,keep]
```

Show kernel log messages that precede the initialization of the traditional console. These messages are typically never seen on the console unless you use this option. Enabling this can be very useful for tracking down hardware issues. Currently, the option can specify either the VGA device or the serial port, but not both at the same time. Also, only the `ttyS0` or `ttyS1` serial devices will work. Interaction with the standard serial driver is not very good, and the VGA output will eventually be overwritten by the real console.

Append `,keep` in order not to disable the messages shown by this option when the real kernel console is initialized and takes over the system.

loglevel

Set the default console log level.

```
loglevel=level
```

Specify the initial console log level. Any log messages with levels less than this (that is, of higher priority) will be printed to the console, whereas any messages with levels equal to or greater than this will not be displayed.

The console log level can also be changed by the *klogd* program, or by writing the specified level to the */proc/sys/kernel/printk* file.

The kernel log levels are:

`0 (KERN_EMERG)`
: The system is unusable.

`1 (KERN_ALERT)`
: Actions that must be taken care of immediately.

`2 (KERN_CRIT)`
: Critical conditions.

`3 (KERN_ERR)`
: Noncritical error conditions.

`4 (KERN_WARNING)`
: Warning conditions that should be taken care of.

`5 (KERN_NOTICE)`
: Normal, but significant events.

`6 (KERN_INFO)`
: Informational messages that require no action.

`7 (KERN_DEBUG)`
: Kernel debugging messages, output by the kernel if the developer enabled debugging at compile time.

log_buf_len

Set the size of the kernel log buffer.

```
log_buf_len=n[KMG]
```

Set the size of the kernel's internal log buffer. *n* must be a power of 2, if not, it will be rounded up to be a power of 2. This value can also be changed by the `CONFIG_LOG_BUF_SHIFT` kernel configuration value.

initcall_debug

Debug the initcall functions in the kernel.

Cause the kernel to trace all functions that are called by the kernel during initialization of the system as the kernel boots. This option is useful for determining where the kernel is dying during startup.

kstack How many words of the stack to print in kernel oopses.

```
kstack=n
```

Specify how many words from the kernel stack should be printed in the kernel oops dumps. *n* is an integer value.

time Show timing data on every kernel log message.

Cause the kernel to prefix every kernel log message with a timestamp.

Interrupt Options

Interrupts are a complex aspect of kernel behavior. The boot time options deal mostly with the interface between the kernel and the hardware that handles interrupts, such as the Intel chip's Advanced Programmable Interrupt Controller (APIC).

apic Change the verbosity of the APIC subsystem when booting.

```
apic=[quiet|verbose|debug]
```

Control how much information the APIC subsystem generates when booting the kernel. The default is `quiet`.

noapic Do not use any IOAPICs.

Prevent the kernel from using any of the IOAPICs that might be present in the system.

lapic Enable the local APIC.

Cause the kernel to enable the local APIC even if the BIOS had disabled it.

nolapic Do not use the local APIC.

Tell the kernel not to use the local APIC.

noirqbalance Disable kernel IRQ balancing.

Disable all of the built-in kernel IRQ balancing logic.

irqfixup

Basic fix to interrupt problems.

When an interrupt is not handled, search all known interrupt handlers for it. This is intended to get systems with badly broken firmware running.

irqpoll

Extended fix to interrupt problems.

When an interrupt is not handled, search all known interrupt handlers for it and also check all handlers on each timer interrupt. This is intended to get systems with badly broken firmware running.

noirqdebug

Disable unhandled interrupt detection.

By default, the kernel attempts to detect and disable unhandled interrupt sources because they can cause problems with the responsiveness of the rest of the kernel if left unchecked. This option disables this logic.

Memory Options

The kernel handles memory in many different chunks and categories for different purposes. These options allow you to tweak the sizes and settings.

highmem

Specify the size of the highmem memory zone.

```
highmem=n
```

Force the highmem memory zone to have an exact size of *n* bytes. This will work even on boxes that have no highmem zones by default. It can also reduce the size of the highmem zone for machines with a lot of memory.

hugepages

Set the number of hugetlb pages.

```
hugepages=n
```

The hugetlb feature lets you configure Linux to use 4 MB pages, one thousand times the default size. If Linux is configured this way, this options sets the maximum number of hugetlb pages to be *n*.

ihash_entries Set the number of inode hash buckets.

```
ihash_entries=n
```

Override the default number of hash buckets for the kernel's inode cache. Recommended only for kernel experts.

max_addr Ignore memory.

```
max_addr=n
```

Cause the kernel to ignore all physical memory greater than or equal to the physical address *n*.

mem Force memory usage.

```
mem=n[KMG]
```

Set the specific ammount of memory used by the kernel. When used with the `memmap=` option, physical address space collisions can be avoided. Without the `memmap=` option, this option could cause PCI devices to be placed at addresses that belong to unused RAM. *n* specifies the amount of memory to force and is measured in units of kilobytes (`K`), megabytes (`M`), or gigabytes (`G`).

mem Disable the use of 4 MB pages for kernel memory.

```
mem=nopentium
```

Disable the use of huge (4 MB) pages for kernel memory.

memmap Enable setting of an exact E820 memory map.

```
memmap=exactmap
```

Use a specific memory map. The *`exactmap`* lines can be constructed based on BIOS output or other requirements.

memmap Force specific memory to be used.

```
memmap=n[KMG]@start[KMG]
```

Force the kernel to use a specific memory region. *n* is the size of the memory location, and *`start`* is the start location in memory of the range. Units can be kilobytes (`K`), megabytes (`M`), or gigabytes (`G`).

noexec Enable or disable nonexecutable mappings.

`noexec=[on|off]`

Enable or disable the kernel's ability to map sections of memory as nonexecutable. By default, the mapping is enabled (`on`).

reserve Reserve some I/O memory.

`reserve=`*`n`*`[KMG]`

Force the kernel to ignore some of the I/O memory areas.

vmalloc Force the vmalloc area to have a specific size.

`vmalloc=`*`n`*`[KMG]`

Force *vmalloc* to have the exact size specified by *n*. This can be used to increase the minimum size of the *vmalloc* area (which is 128 MB on the x86 processor). It can also be used to decrease the size and leave more room for directly mapped kernel RAM.

norandmaps Do not use address space randomization.

By default, the kernel randomizes the address space of all programs when they are started. This option disables this feature. It is equivalent to writing `0` to the file */proc/sys/kernel/randomize_va_space*.

vdso Enable or disable the VDSO mapping.

`vdso=[0|1]`

Disable (`0`) or enable (`1`) the VDSO (Virtual Dynamic Shared Object) mapping option. By default, it is enabled.

Suspend Options

These options change the way the kernel handles suspension for power-saving purposes.

resume Specify the partition device for the suspend image.

`resume=`*`suspend_device`*

Tell the kernel which disk device contains the suspended kernel image. If the data on the image is a valid kernel image created by the software suspend subsystem, it will be loaded into memory and the kernel will run it instead of continuing on with the normal boot process. *suspend_device* is the kernel device name, which might be different from what userspace thinks the device name is, so be careful with this option.

noresume Disable resume.

Disable the resume functionality of the kernel. Any swap partitions that were being used to hold system images to which the kernel could be restored will revert back to available swap space.

CPU Options

These options control a wide range of behavior regarding timing, processor use in multiprocessor systems, and other processor issues.

cachesize Override level 2 CPU cache size detection.

```
cachesize=n
```

Sometimes CPU hardware bugs make them report the cache size incorrectly. The kernel will attempt to work around and fix known problems with most CPUs, but for some CPUs it is not possible to determine what the correct size should be. This option provides an override for these situations. *n* is measured in bytes.

lpj Set the loops per jiffy.

```
lpg=n
```

Specify the loops per jiffy that should be used by the kernel, and thus have the kernel avoid the time-consuming boot-time autodetection of this value. If *n* is `0`, the value will be autodetected as usual.

On SMP systems, this value will be set on all CPUs, which might cause problems if the different CPUs need different settings. An incorrect value will cause incorrect delays in the kernel, which can lead to unpredictable I/O errors and other breakage. Although unlikely, in extreme cases this might damage your hardware.

nmi_watchdog

Set the NMI watchdog value.

```
nmi_watchdog=[0|1|2|3]
```

This is a debugging feature that allows the user to override the default nonmaskable interrupt (NMI) watchdog value. `0` specifies that no NMI watchdog should be used. `1` specifies that the APIC should be used if present. `2` specifies that the local APIC should be used if present. `3` means that the NMI watchdog is invalid, so do not use it.

no387

Always use the 387 emulation library.

Always use the 387 math emulation library, even if a 387 math coprocessor is present in the system.

nofxsr

Disable x86 floating-point save and restore.

Disable the x86 floating-point extended register save and restore. The kernel will save only legacy floating-point registers on a task switch.

no-hlt

Do not use the `HLT` instruction.

This option is available because the `HLT` instruction does not work correctly for some x86 processors. This option tells the kernel not to use the instruction.

mce

Enable the machine check exception feature.

Some processors can check for machine errors (usually errors in the hardware). This option turns this subsystem on, if it has been built into the kernel configuration.

nomce

Disable the machine check exception feature.

This option turns the subsystem off.

nosep

Disable x86 SYSENTER/SYSEXIT support.

Disable x86 SYSENTER/SYSEXIT support in the kernel. This can cause some system calls to take longer.

nosmp

Run as a single-processor machine.

Tell an SMP kernel to act as a uniprocessor kernel, even on a multiprocessor machine.

notsc Disable the time stamp counter.

Disable the timestamp counter hardware in the system, if present.

max_cpus Maximum number of CPUs to use.

```
maxcpus=n
```

Specify the maximum number of processors that a SMP kernel should use, even if there are more processors present in the system.

Scheduler Options

These options tweak the parameters used to make scheduling decisions. Most depend on an intimate understanding of how scheduling works in Linux.

isolcpus Isolate CPUs from the kernel scheduler.

```
isolcpus=cpu_number[,cpu_number,...]
```

Remove the specified CPUs, as defined by the *cpu_number* values, from the general kernel SMP balancing and scheduler algroithms. The only way to move a process onto or off an “isolated” CPU is via the CPU affinity syscalls. *cpu_number* begins at `0`, so the maximum value is one less than the number of CPUs on the system.

This option is the preferred way to isolate CPUs. The alternative, manually setting the CPU mask of all tasks in the system, can cause problems and suboptimal load-balancer performance.

migration_cost Override the default scheduler migrations costs.

```
migration_cost=level-1-useconds[level-2-useconds...]
```

This is a debugging option that overrides the default scheduler migration cost matrix. The numbers specified by *level-N-useconds* are indexed by the “CPU domain distance” and are measured in microseconds.

An example of this option is `migration_cost=1000,2000,3000` for a SMT NUMA machine. It sets up an intra-core migration cost of 1 ms, another inter-core migration cost of 2 ms, and another inter-node migration cost of 3 ms.

Incorrect values can severely degrade scheduler performance, so this option should be used only for scheduler development, never for production environments.

migration_debug

Verbosity of migration cost autodetection.

```
migration_debug=[0|1|2]
```

Set the migration cost debug level. If `0` is specified, no extra messages will be printed to the kernel log. This is the default value. `1` prints some information on how the matrix is determined. `2` is very verbose and is useful only if you use a serial console, as the amount of information will overflow the kernel log buffer.

migration_factor

Multiply or divide the migration costs.

```
migration_factor=percent
```

Modify the default migration costs by the specified *`percent`*. This is a debugging option that can be used to proportionally increase or decrease the autodetected migration costs for all entries of the migration matrix. For example, `migration_factor=150` increases migration costs by 50 percent, so the scheduler will be less eager to migrate cache-hot tasks. `migration_factor=80` decreases migration costs by 20 percent, thus making the scheduler more eager to migrate tasks.

> Incorrect values can severely degrade scheduler performance, so this option should be used only for scheduler development, never for production environments.

Ramdisk Options

These options control how the storage of information in memory used to imitate disks (ramdisks) is done, including init ramdisks that hold information necessary at some stages of booting.

initrd

Location of initial ramdisk.

```
initrd=filename
```

Specify where the initial ramdisk for the kernel boot is located.

load_ramdisk

Load a kernel ramdisk from a floppy.

```
load_ramdisk=n
```

If *`n`* is set to `1`, a ramdisk is loaded by the kernel at boot time from the floppy drive.

noinitrd

Do not use any initrd.

Do not load any initial ramdisk, even if it is configured in other options passed to the kernel.

prompt_ramdisk

Prompt for the list of ramdisks.

```
prompt_ramdisk=1
```

Prompt the user for the initial ramdisk before attempting to read it from the floppy drive.

ramdisk_blocksize

Blocksize of the ramdisk.

```
ramdisk_blocksize=n
```

Tell the ramdisk driver how many bytes to use per block. The default size is 1,024.

ramdisk_size

Size of the ramdisk.

```
ramdisk_size=n
```

Specify the size of the initial ramdisk in kilobytes. The default size is 4,096 (4 MB). This option should be used instead of the older *ramdisk* command-line option.

Root Disk Options

These options control how the kernel finds and handles the filesystem that contains the root filesystem.

ro

Mount the root device read-only on boot.

The default for the kernel is to mount the root device as read-only at boot time. This option ensures that this is the mode the kernel uses. It overrides the `rw` command-line option, if it had been specified earlier on the boot command line.

root

Specify the root filesystem to boot from.

```
root=device
```

Tell the kernel which disk device the root filesystem image is on. *`device`* can be specified in one of the following ways:

`nnnn`
: A device number in hexadecimal represents the major and minor number of the device in the internal format that the kernel expects. This method is not recommended unless you have access to kernel internals.

`/dev/nfs`
: Use the NFS disk specified by the `nfsroot` boot option as the root disk.

`/dev/<diskname>`
: Use the kernel disk name specified by *`<diskname>`* as the root disk.

`/dev/<diskname><decimal>`
: Use the kernel disk name specified by *`<diskname>`* and the partition specified by *`<decimal>`* as the root disk.

`/dev/<diskname>p<decimal>`
: Use the kernel disk name specified by *`<diskname>`* and the partition specified by *`<decimal>`* as the root disk. This is the same as above, but is needed when *`<diskname>`* ends with a digit.

rootdelay

Time to delay before attempting to mount the root filesystem.

```
rootdelay=n
```

Wait *n* seconds before trying to mount the root filesystem. This can be useful if the root filesystem is on a USB or FireWire device, as those disk devices take a bit longer to be discovered by the kernel.

rootflags

The root filesystem mount options.

```
rootflags=options
```

Mount options that the kernel should use in mounting the root filesystem. The *`options`* value depend on the filesystem type; see the documentation for the individual types for details on what is valid.

rootfstype

The root filesystem type.

```
rootfstype=type
```

Try to mount the root filesystem as this type of filesystem. For instance, `rootfstype=ext3`.

rw

Mount the root device read-write on boot.

The default for the kernel is to mount the root device as read-only at boot time. This option mounts the root device as read-write instead.

Init Options

The *init* process is the first to be started by the kernel and is the ancestor of all other processes. These options control which program is run and how it is run.

---

init Program to run at init time.

```
init=filename
```

Run the specified binary as the *init* process instead of the default */sbin/init* program.

---

rdinit Run the *init* process from the ramdisk.

```
rdinit=full_path_name
```

Run the program specified by `full_path_name` as the *init* process. This file must be on the kernel ramdisk instead of on the root filesystem.

S Run *init* in single-user mode.

The default for the kernel is to run *init* in multi-user mode. This option runs *init* in single-user mode instead.

kexec Options

The kexec subsystem is a specialized rebooting feature that allows a fast reboot and is usually combined with the kdump facility that enables the previous kernel's memory to be dumped to a safe place for analysis at a later time. These options modify the kexec subsystem's parameters.

---

crashkernel Reserve a portion of physical memory for kexec to use.

```
crashkernel=n[KMG]@start[KMG]
```

The kexec subsystem likes to have a portion of physical memory reserved for it. This option reserves that memory from the rest of the kernel and will switch to use it if the kernel panics. `n` specifies the amount of memory to reserve, and `start` specifies the location for this memory chunk. Both are measured in units of kilobytes (`K`), megabytes (`M`), or gigabytes (`G`).

elfcorehdr Start of the kernel core image ELF header.

```
elfcorhdr=n
```

The kernel, like every Linux executable, is stored in ELF format. This option specifies the physical address where the kernel core image's ELF header starts. This is used by kexec to find the kernel when booting the secondary kernel image.

RCU Options

Read Copy Update (RCU) is a portion of the kernel that handles mutual exclusion for a variety of subsystems in a lockless manner. There are a number of options that can be used to tune RCU in different ways:

rcu.blimit RCU batch limit.

```
rcu.blimit=n
```

Set the maximum number of finished RCU callbacks to process in one batch.

rcu.qhimark RCU queue high level.

```
rcu.qhimark=n
```

Batch limiting is disabled when the number of queued RCU callbacks rises above *n*.

rcu.qlowmark RCU queue low level.

```
rcu.qlowmark=n
```

Batch limiting is re-enabled when the number of queued RCU callbacks falls below *n*.

rcu.rsinterval RCU callback queue length.

```
rcu.rsinterval=n
```

Set the number of additional RCU callbacks that should be queued before forcing a reschedule on all CPUs.

ACPI Options

These options control parameters that the Advanced Configuration and Power Interface (ACPI) subsystem can use.

acpi

ACPI subsystem options.

```
acpi=[force|off|noirq|ht|strict]
```

This is the main option for the Advanced Configuration and Power Interface (ACPI). Values are:

`force`
: Force ACPI to be enabled. Can be used to override the kernel configuration option that disabled it.

`off`
: Disable ACPI. Can be used to override the kernel configuration option that enabled it.

`noirq`
: Prevent ACPI from being used for IRQ routing.

`ht`
: Run only enough of the ACPI layer to enable HyperThreading on processors that are capable of it.

`strict`
: Make the ACPI layer be less tolerant of platforms that are not fully compliant with the ACPI specification.

acpi_sleep

ACPI sleep options.

```
acpi_sleep=[s3_bios],[s3_mode]
```

During S3 resume (which happens after the machine has been suspended to RAM), hardware needs to be reinitialized properly. For most devices this is simple, except for video cards, which are normally initialized by the BIOS. The kernel does not have enough information to restore the video device, because that information is in the BIOS and not accessable at all. This option lets the kernel try to use the ACPI subsystem to restore the video card in two different ways.

See the file *Documentation/power/video.txt* for more information on this option and how to find the proper value for your type of hardware.

acpi_sci

ACPI System Control Interrupt trigger mode.

```
acpi_sci=[level|edge|high|low]
```

Set the ACPI System Control Interrupt trigger mode.

acpi_irq_balance

Enable ACPI IRQ balance.

Cause ACPI to balance the active IRQs. This is the default option when operating in APIC mode.

acpi_irq_nobalance

Disable ACPI IRQ balance.

Cause ACPI not to move the active IRQs. This is the default option when operating in PIC mode.

acpi_irq_isa

Mark the listed IRQs as used by ISA.

```
acpi_irq_isa=irq[,irq...]
```

If the IRQ balance option is enabled, mark the listed IRQs as used by the ISA subsystem.

acpi_irq_pci

Mark the listed IRQs as used by PCI.

```
acpi_irq_pci=irq[,[irq...]
```

If the IRQ balance option is enabled, mark the listed IRQs as used by the PCI subsystem.

acpi_os_name

Fake the operating system name to ACPI.

```
acpi_os_name=name
```

Tell the ACPI BIOS that the name of the running operating system is *name*. This can be useful to spoof the BIOS into thinking that Windows is running instead of Linux, which can help solve some ACPI issues for older BIOSes. As an example, use the string `Microsoft 2001` to spoof the BIOS into thinking that Windows 2001 is running on the machine.

acpi_osi

Disable the _OSI ACPI method.

```
acpi_osi=[n]
```

This is actually a binary option despite the integer value. If *n* is absent, ACPI will disable the _OSI method. If *n* is present, _OSI will not be disabled.

acpi_serialize

Force serialization of AML methods.

Force the serialization of ACPI Machine Language methods.

acpi_skip_ timer_override	Skip interrupt override issues. Allow the ACPI layer to recognize and ignore IRQ0/pin2 interrupt override issues for broken nForce2 BIOSes that result in the XT-PIC timer acting up.
acpi_dbg_layer	ACPI debug layer. `acpi_dbg_layer=`*`n`* Set the ACPI debug layers. *n* is an integer in which each bit indicates a different ACPI debug layer. After the system has booted, the debug layers can be set via the */proc/acpi/debug_layer* file.
acpi_fake_ecdt	ECDT workaround. If present, this allows ACPI to workaround BIOS failures when it lacks an Embedded Controller Description Table.
acpi_generic_ hotkey	Use generic ACPI hotkey driver. This allows the ACPI consolidated generic hotkey driver to override the platform-specific driver if one is present.
acpi_pm_good	Override pmtimer bug detection. Force the kernel to assume that the machine's pmtimer latches its value and always returns good values.
ec_intr	ACPI Embedded Controller interrupt mode. `ec_intr=`*`n`* Specify the ACPI embedded controller interrupt mode. If *n* is `0`, polling mode will be used, otherwise interrupt mode will be used. Interrupt mode is the default.
memmap	Mark specific memory as ACPI data. `memmap=`*`n`*`[KMG]#`*`start`*`[KMG]` Marks a specific location and range of memory as ACPI data. *n* is the size of the memory location and *start* is the start location in memory of the range. Both are measured in units of kilobytes (`K`), megabytes (`M`), or gigabytes (`G`).

memmap

Mark specific memory as reserved.

```
memmap=n[KMG]$start[KMG]
```

This marks a specific location and range of memory as reserved. *n* is the size of the memory location and *start* is the start location in memory of the range.

pnpacpi

Turn Plug and Play ACPI off.

```
pnpacpi=off
```

Disable the Plug and Play ACPI functionality.

processor.max_cstate

Limit the processor to a maximum C-state.

```
processor.max_cstate=n
```

Limit the processor to a maximum C-state, no matter what the ACPI tables say it can support. *n* is a valid C-state value. A value of `9` overrides any DMI blacklist limit that might be present for this processor.

processor.nocst

Ignore the _CST method for C-states.

Causes the ACPI core to ignore the _CST method of determining the processor C-states and use the legacy FADT method instead.

SCSI Options

These options specify different parameters the SCSI subsystem can use. A number of SCSI driver-specific options are also available; please see the different driver documentation files in the kernel directory *Documentation/scsi/* for details.

max_luns

Maximum number of SCSI LUNS to probe.

```
max_luns=n
```

Specify the maximum number of SCSI LUNS that the system should probe. *n* is an integer from 1 to 4,294,967,295.

max_report_luns

Maximum number of SCSI LUNS received.

```
max_report_luns=n
```

Specify the maximum number of SCSI LUNs that the system can receive. *n* is an integer from 1 to 16,384.

scsi_dev_flags SCSI black/white list.

```
scsi_dev_flags=vendor:model:flags
```

This option lets the user add entries to the SCSI black/white list for a specific vendor and model of device.

PCI Options

These options specify different parameters the PCI subsystem can use:

PCI

```
pci=option[,option...]
```

Each *option* can be one of the following:

`off`
: Do not probe for the PCI bus.

`bios`
: Force the use of the PCI BIOS by not accessing the hardware directly. This means that the kernel should trust the BIOS, which is not the standard thing to do (as BIOSes are known to lie more often than they are known to be valid). Use this only if your machine has a nonstandard PCI host bridge and the normal boot method is not working properly.

`nobios`
: Do not use the PCI BIOS, but access the hardware directly instead. This is the default method of probing for PCI devices in all kernels after 2.6.13.

`conf1`
: Force use of PCI Configuration Mechanism 1 (a way to access PCI memory on i386 machines).

`conf2`
: Force use of PCI Configuration Mechanism 2 (a way to access PCI memory on i386 machines).

`nommconf`
: Disable use of the ACPI MMCONFIG table for PCI configuration.

`nomsi`
: If the PCI_MSI kernel config parameter is enabled, this kernel boot option can be used to disable the use of MSI interrupts system-wide.

`nosort`
: Do not sort PCI devices according to order given by the PCI BIOS. This sorting is done to get a device order compatible with much older kernel versions.

`biosirq`
: Use PCI BIOS calls to get the interrupt routing table. These calls are known to be buggy on several machines and hang these machine when used, but on other machines they are the only way to get the interrupt routing table. Try this option if the kernel is unable to allocate IRQs or discover secondary PCI buses on your motherboard.

`rom`
: Assign address space to expansion ROMs. Use this with caution as certain devices share address decoders between ROMs and other resources.

`irqmask=0x`*`nnnn`*
: Set a bit mask of IRQs allowed to be assigned automatically to PCI devices. You can make the kernel exclude IRQs of your ISA cards this way.

`pirqaddr=0x`*`n`*
: Specify the physical address of the PIRQ table (normally generated by the BIOS) if it is outside the F0000–100000 (hexadecimal) range.

`lastbus=`*`n`*
: Scan all buses through bus *n*. Can be useful if the kernel is unable to find your secondary buses and you want to tell it explicitly which ones they are.

`assign-busses`
: Always use your own PCI bus numbers, overriding whatever the firmware may have done.

`usepirqmask`
: Honor the possible IRQ mask stored in the BIOS $PIR table. This is needed on some systems with broken BIOSes, notably some HP Pavilion N5400 and Omnibook XE3 notebooks. This will have no effect if ACPI IRQ routing is enabled.

`noacpi`
: Do not use ACPI for IRQ routing or for PCI scanning.

`routeirq`
: Do IRQ routing for all PCI devices. This is normally done in `pci_enable_device()`, so this option is a temporary workaround for broken drivers that don't call it.

`firmware`
: Do not re-enumerate the bus, but instead just use the configuration from the bootloader. This is currently used on IXP2000 systems where the bus has to be configured a certain way for adjunct CPUs.

Plug and Play BIOS Options

noisapnp Disable the ISA Plug and Play (PnP) subsystem.

Disable the ISA PnP subsystem, if it has been enabled in the kernel configuration.

pnpbios PnP BIOS settings.

```
pnpbios=[on|off|curr|no-curr]
```

Set the main PnP BIOS settings. `on` enables the PnP BIOS subsystem. `off` disables the PnP BIOS subsystem. `curr` tells the PnP BIOS subsystem to use the current static settings and `no-curr` tells the subsystem to probe for dynamic settings if possible.

pnp_reserve_irq PnP BIOS reserved IRQs.

```
pnp_reserve_irq=irq1[,irq2...]
```

List of the IRQs that the PnP BIOS subsystem should not use for autoconfiguration.

pnp_reserve_dma PnP BIOS reserved DMAs.

```
pnp_reserve_dma=dma1[,dma2...]
```

List of the DMAs that the PnP BIOS subsystem should not use for autoconfiguration.

pnp_reserve_io PnP BIOS reserved I/O ports.

```
pnp_reserve_io=io1,size1[,io2,size2...]
```

I/O ports that the PnP BIOS subsystem should not use for autoconfiguration. Each port is listed by its starting location and size.

pnp_reserve_mem PnP BIOS reserved memory regions.

```
pnp_reserve_mem=mem1,size1[,mem2,size2...]
```

Memory regions that the PnP BIOS subsystem should not use for autoconfiguration. Each region is listed by its starting location and size.

SELinux Options

These options change some fundamental aspects of SELinux startup.

checkreqprot

Set the initial checkreqprot flag value.

```
checkreqprot=[0|1]
```

Set the initial *checkreqprot* flag value. 0 means that the check protection will be applied by the kernel and will include any implied execute protection. 1 means that the check protection is requested by the application. The default value is set by a kernel configuration option.

The value can be changed at runtime via the */selinux/checkreqprot* file.

enforcing

Set the initial enforcing status.

```
enforcing=[0|1]
```

Specify whether SELinux enforces its rules upon boot. 0 means that SELinux will just log policy violations but will not deny access to anything. 1 means that the enforcement will be fully enabled with denials as well as logging. The default value is 0.

The value can be changed at runtime via the */selinux/enforce* file.

selinux

Enable or disable SELinux at boot time.

```
selinux=[0|1]
```

This option allows SELinux to be enabled (1) or disabled (0) to boot time. The default value is set by a kernel configuration option.

If SELinux is enabled at boot time, the */selinux/disable* file can be used later to disable it prior to the initial policy load.

selinux_compat_net

Set the network control model.

```
selinux_compat_net=[0|1]
```

Set the initial value for the SELinux network control model. 0 uses the new secmark-based packet controls, and 1 uses the legacy packet controls. 0 is the default and preferred value.

This value can be changed at runtime via the */selinux/compat_net* file.

Network Options

These options control low-level aspects of the networking subsystem.

netdev Set various network device parameters.

netdev=[*irq*],[*io*],[*mem_start*],[*mem_end*],[*name*]

Specify network device parameters, which are specific to the driver used by the network device. Some drivers' source files document the applicable options. This option does not usually apply to PCI, USB, or other plug-and-play network devices. It is intended for use only on devices that can not discover their own resource assignments.

rhash_entries Set the number of route cache hash buckets.

dhash_entries=*n*

This option lets you override the default number of hash buckets for the kernel's route cache. Recommended only for kernel network experts.

shapers Set the maximum number of network shapers.

shapers=*n*

This option lets you set the maximum number of network shapers that the kernel can use.

thash_entries Set the number of TCP connection hash buckets.

thash_entries=*n*

This option lets you override the default number of hash buckets for the kernel's TCP connection cache.

Network File System Options

These options control NFS startup.

lockd.nlm_ grace_period Assign a grace period to the lock manager.

lockd.nlm_grace_period=*n*

Set the NFS lock manager grace period. *n* is measured in seconds.

lockd.nlm_tcpport

Assign a TCP port to the lock manager.

```
lockd.nlm_tcpport=port
```

Set the TCP port that the NFS lock manager should use. `port` must be a valid TCP port value.

lockd.nlm_timeout

Assign a new timeout value to the lock manager.

```
lockd.nlm_timeout=n
```

Override the default time value for the NFS lock manager. *n* is measured in seconds. If this option is not specified, the default of 10 seconds will be used.

lockd.nlm_udpport

Assign a UDP port to the lock manager.

```
lockd.nlm_udpport=port
```

Set the UDP port that the NFS lock manager should use. `port` must be a valid UDP port value.

nfsroot

Specifies the NFS root filesystem.

```
nfsroot=[server-ip:]root-dir[,nfs-options]
```

Set the NFS root filesystem for diskless boxes, to enable them to boot properly over NFS. If this parameter is not set, the value */tftpboot/client_ip_address* will be used as the root filesystem with the default NFS options.

server-ip
: IP address of the NFS server to connect to.

root-dir
: Directory on the NFS server to mount as root. If there is a `%s` token in this string, it will be replaced with the ASCII representation of the client's IP address.

nfs-options
: The standard NFS options, such as `ro`, separated by commas.

nfs.callback_tcpport

Set the NFSv4 TCP port for the callback channel.

```
nfs.callback_tcpport=port
```

Specify the TCP port that the NFSv4 callback channel should listen on. `port` must be a valid TCP port value.

nfs.idmap_cache_timeout

Set the maximum lifetime for idmapper cache entries.

```
nfs.idmap_cache_timeout=n
```

Specify the maximum lifetime for idmapper cache entries. *n* is measured in seconds.

Hardware-Specific Options

These options specify different parameters, depending on the hardware present in the system.

nousb

Disable the USB subsystem.

If this option is present, the USB subsystem will not be initialized.

lp

Parallel port and its mode.

```
lp=[0|port[,port...]|reset|auto]
```

Specify the parallel port to use. The `lp=port1,port2...` format associates a sequence of parallel ports to devices, starting with `lp0`. An example is `lp=none,parport0`, which would suppress configuration of the `lp0` device and cause the `lp1` device to use the first parallel port.

`lp=0`
: Disables the printer driver.

`lp=reset`
: Causes the attached printers to be reset. This option can be combined with the port specifications.

`lp=auto`
: Causes the kernel to examine the device ID from each port to determine whether a IEEE 1284-compatible printer is attached. If so, the kernel will manage that printer.

parport

Specify the parallel port parameters.

```
parport=[setting[,setting...]
```

Specify settings for parallel port drivers. Parallel ports are assigned in the order they are specified on the command line, starting with `parport0`.

`auto` forces the driver to use any IRQ/DMA settings detected (the default is to ignore detected IRQ/DMA settings because of possible conflicts). You can also specify the base address, IRQ, and DMA settings in the format `0x`*nnnn*[,*irq*[,*dma*]]. *irq* and *dma* can be numbers, `auto` to use detected settings on that particular port, or `nofifo` to avoid using a FIFO even if it is detected.

parport_init_mode

Parallel port initialization mode.

```
parport_init_mode=[spp|ps2|epp|ecp|ecpepp]
```

Specifies the mode for operating the parallel port. This is necessary on the Pegasos computer where the firmware has no options for setting up the parallel port mode. This option works for parallel port chips of type 686a and 8231.

nr_uarts

Maximum number of UARTs to be registered.

```
nr_uarts=n
```

Specifies the maximum number of different UARTs that can be registered in the kernel.

Timer-Specific Options

These options override default kernel behavior to fix problems with certain chips.

enable_timer_pin_1

Enable pin 1 of the APIC timer.

Enable pin 1 of the APIC timer. This option can be useful to work around chipset bugs (on some ATI chipsets in particular). The kernel tries to set a reasonable default, but sometimes this option is necessary to override it.

disable_timer_pin_1

Disable pin 1 of the APIC timer.

Disable pin 1 of the APIC timer. Useful for the same reasons as `enable_timer_pin_1`.

enable_8254_timer

Enable interrupt 0 timer routing over the 8254 chip.

Enable interrupt 0 timer routing over the 8254 chip in addition to routing over the IO-APIC. The kernel tries to set a reasonable default, but sometimes this option is necessary to override it.

disable_8254_timer

Disable interrupt 0 timer routing over the 8254 chip.

Disable interrupt 0 timer routing over the 8254 chip in addition to routing over the IO-APIC. The kernel tries to set a reasonable default, but sometimes this option is necessary to override it.

hpet

Disable HPET and use PIT instead.

```
hpet=disable
```

Disable the HPET timer source and tell the kernel to use the PIT timer source instead.

clocksource

Set the specific clocksource.

```
clocksource=[hpet|pit|tsc|acpi_pm|cyclone|scx200_hrt]
```

Override the default kernel clocksource and use the clocksource with the specified name instead.

Miscellaneous Options

These options should always be available and don't depend on any specific subsystem or hardware being present in the system in order to work properly.

dhash_entries

Set the number of dentry hash buckets.

```
dhash_entries=n
```

This option lets you override the default number of hash buckets for the kernel's dentry cache. Recommended only for kernel experts.

elevator

Set the default I/O scheduler elevator.

```
elevator=[anticipatory|cfq|deadline|noop]
```

Specify the I/O scheduler. See Chapter 11 for a list of the different I/O schedulers available, and what they do.

hashdist

Distribute large hashes across NUMA nodes.

```
hashdist=[0|1]
```

Large hashes that are allocated during the boot process on the IA-64 platform are, by default, distributed across the different NUMA nodes. This option lets the user turn this option on or off.

combined_mode Specify IDE driver usage.

```
combined_mode=[combined|ide|libata]
```

Control which driver uses the IDE ports in combined mode: the legacy IDE driver, *libata*, or both. Note that using the `ide` or `libata` options may affect your device naming (e.g., by changing `hdc` to `sdb`).

max_loop Maximum number of loopback devices.

```
max_loop=n
```

Specify the maximum number of loopback filesystem devices that can be mounted at the same time. *n* is an integer from 1 to 256.

panic Time to wait after panic before rebooting.

```
panic=n
```

Specify the amount of time in seconds that the kernel should wait after a panic happens before it reboots. If this is set to `0` (the default value), the kernel will not reboot after panicking; it will simply halt.

pause_on_oops Delay between kernel oopses.

```
pause_on_oops=n
```

Tell the kernel to halt all CPUs after the first oops for *n* seconds before continuing. This is useful if oopses keep scrolling off of the screen before you can write them down or take a picture of them.

profile Control the kernel profiling.

```
profile=[schedule,][number]
```

This option affects how the kernel profiler is calculated. If `schedule` is specified, the schedule points are affected by the value set in *`number`*. If *`schedule`* is not specified, *`number`* is the step size as a power of two for statistical time-based profiling in the kernel.

The most common use of this option is `profile=2`.