



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.

|                   |  |
|-------------------|--|
| <b>console</b>    | Output console device and options.<br><br>console=Options<br><br>ttn<br>Use the virtual console device n.<br><br>ttySn[,options], ttyUSB0[,options]<br>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.<br><br>See the file <i>Documentation/serial-console.txt</i> 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 <i>netconsole</i> command-line option.<br><br>uart,io,addr[,options], uart,mmio,addr[,options]<br>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. |
| <b>netconsole</b> | Output console data across the network.<br><br>netconsole=[src-port]@[src-ip]/[dev],[target-port]@target-ip/[target-mac-address]<br><br>Send kernel console data across the network using UDP packets to another machine. Options are:   |



## 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.



---

**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`.



|                   |   |
|-------------------|---|
| <b>noexec</b>     | <p>Enable or disable nonexecutable mappings.</p> <p><code>noexec=[on off]</code></p> <p>Enable or disable the kernel's ability to map sections of memory as nonexecutable. By default, the mapping is enabled (on).</p>   |
| <b>reserve</b>    | <p>Reserve some I/O memory.</p> <p><code>reserve=n[KMG]</code></p> <p>Force the kernel to ignore some of the I/O memory areas.</p>  |
| <b>vmalloc</b>    | <p>Force the vmalloc area to have a specific size.</p> <p><code>vmalloc=n[KMG]</code></p> <p>Force <i>vmalloc</i> to have the exact size specified by <i>n</i>. This can be used to increase the minimum size of the <i>vmalloc</i> 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.</p> |
| <b>norandmaps</b> | <p>Do not use address space randomization.</p> <p>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 <code>/proc/sys/kernel/randomize_va_space</code>.</p>   |
| <b>vdso</b>       | <p>Enable or disable the VDSO mapping.</p> <p><code>vdso=[0 1]</code></p> <p>Disable (0) or enable (1) the VDSO (Virtual Dynamic Shared Object) mapping option. By default, it is enabled.</p>  |

## Suspend Options

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

|               |  |
|---------------|--|
| <b>resume</b> | <p>Specify the partition device for the suspend image.</p> <p><code>resume=suspend_device</code></p> <p>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</p> |
|---------------|--|





## 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 multi-processor machine.



---

**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.



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.



---

**elfcorehdr**

Start of the kernel core image ELF header.

`elfcorehdr=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.

---





|                           |  |
|---------------------------|--|
| <b>acpi_irq_balance</b>   | <p>Enable ACPI IRQ balance.</p> <p>Cause ACPI to balance the active IRQs. This is the default option when operating in APIC mode.</p>  |
| <b>acpi_irq_nobalance</b> | <p>Disable ACPI IRQ balance.</p> <p>Cause ACPI not to move the active IRQs. This is the default option when operating in PIC mode.</p>   |
| <b>acpi_irq_isa</b>       | <p>Mark the listed IRQs as used by ISA.</p> <p><code>acpi_irq_isa=irq[,irq...]</code></p> <p>If the IRQ balance option is enabled, mark the listed IRQs as used by the ISA subsystem.</p>  |
| <b>acpi_irq_pci</b>       | <p>Mark the listed IRQs as used by PCI.</p> <p><code>acpi_irq_pci=irq[,irq...]</code></p> <p>If the IRQ balance option is enabled, mark the listed IRQs as used by the PCI subsystem.</p>  |
| <b>acpi_os_name</b>       | <p>Fake the operating system name to ACPI.</p> <p><code>acpi_os_name=name</code></p> <p>Tell the ACPI BIOS that the name of the running operating system is <code>name</code>. 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 <code>Microsoft 2001</code> to spoof the BIOS into thinking that Windows 2001 is running on the machine.</p> |
| <b>acpi_osi</b>           | <p>Disable the <code>_OSI</code> ACPI method.</p> <p><code>acpi_osi=[n]</code></p> <p>This is actually a binary option despite the integer value. If <code>n</code> is absent, ACPI will disable the <code>_OSI</code> method. If <code>n</code> is present, <code>_OSI</code> will not be disabled.</p>   |
| <b>acpi_serialize</b>     | <p>Force serialization of AML methods.</p> <p>Force the serialization of ACPI Machine Language methods.</p>  |



|                             |   |
|-----------------------------|---|
| <b>memmap</b>               | <p>Mark specific memory as reserved.</p> <p><code>memmap=n[KMG]\$start[KMG]</code></p> <p>This marks a specific location and range of memory as reserved. <code>n</code> is the size of the memory location and <code>start</code> is the start location in memory of the range.</p>  |
| <b>pnpcpi</b>               | <p>Turn Plug and Play ACPI off.</p> <p><code>pnpcpi=off</code></p> <p>Disable the Plug and Play ACPI functionality.</p>   |
| <b>processor.max_cstate</b> | <p>Limit the processor to a maximum C-state.</p> <p><code>processor.max_cstate=n</code></p> <p>Limit the processor to a maximum C-state, no matter what the ACPI tables say it can support. <code>n</code> is a valid C-state value. A value of 9 overrides any DMI blacklist limit that might be present for this processor.</p> |
| <b>processor.nocst</b>      | <p>Ignore the <code>_CST</code> method for C-states.</p> <p>Causes the ACPI core to ignore the <code>_CST</code> method of determining the processor C-states and use the legacy FADT method instead.</p>   |

## 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.

|                        |  |
|------------------------|--|
| <b>max_luns</b>        | <p>Maximum number of SCSI LUNS to probe.</p> <p><code>max_luns=n</code></p> <p>Specify the maximum number of SCSI LUNS that the system should probe. <code>n</code> is an integer from 1 to 4,294,967,295.</p> |
| <b>max_report_luns</b> | <p>Maximum number of SCSI LUNS received.</p> <p><code>max_report_luns=n</code></p> <p>Specify the maximum number of SCSI LUNs that the system can receive. <code>n</code> is an integer from 1 to 16,384.</p>  |



#### `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=0xnnnn`

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=0xn`

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 SPIR 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.



## SELinux Options

These options change some fundamental aspects of SELinux startup.

|                           |   |
|---------------------------|---|
| <b>checkreqprot</b>       | <p>Set the initial <i>checkreqprot</i> flag value.</p> <p><code>checkreqprot=[0 1]</code></p> <p>Set the initial <i>checkreqprot</i> 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.</p> <p>The value can be changed at runtime via the <i>/selinux/checkreqprot</i> file.</p> |
| <b>enforcing</b>          | <p>Set the initial enforcing status.</p> <p><code>enforcing=[0 1]</code></p> <p>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.</p> <p>The value can be changed at runtime via the <i>/selinux/enforce</i> file.</p>   |
| <b>selinux</b>            | <p>Enable or disable SELinux at boot time.</p> <p><code>selinux=[0 1]</code></p> <p>This option allows SELinux to be enabled (1) or disabled (0) to boot time. The default value is set by a kernel configuration option.</p> <p>If SELinux is enabled at boot time, the <i>/selinux/disable</i> file can be used later to disable it prior to the initial policy load.</p>   |
| <b>selinux_compat_net</b> | <p>Set the network control model.</p> <p><code>selinux_compat_net=[0 1]</code></p> <p>Set the initial value for the SELinux network control model. 0 uses the new <i>secmark</i>-based packet controls, and 1 uses the legacy packet controls. 0 is the default and preferred value.</p> <p>This value can be changed at runtime via the <i>/selinux/compat_net</i> file.</p>   |





**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 */tftp-boot/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.



conflicts). You can also specify the base address, IRQ, and DMA settings in the format `0xnnnn[,irq[,dma]]`. `irq` and `dma` can be numbers, `auto` to use detected settings on that particular port, or `no fifo` 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.



---

**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`.

---