Linux Kernel User Documentation

Release

The kernel development community

Nov 02, 2017

CONTENTS

The following is a collection of user-oriented documents that have been added to the kernel over time. There is, as yet, little overall order or organization here — this material was not written to be a single, coherent document! With luck things will improve quickly over time.

This initial section contains overall information, including the README file describing the kernel as a whole, documentation on kernel parameters, etc.

CHAPTER

ONE

LINUX KERNEL RELEASE 4.X <HTTP://KERNEL.ORG/>

These are the release notes for Linux version 4. Read them carefully, as they tell you what this is all about, explain how to install the kernel, and what to do if something goes wrong.

What is Linux?

Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It aims towards POSIX and Single UNIX Specification compliance.

It has all the features you would expect in a modern fully-fledged Unix, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multistack networking including IPv4 and IPv6.

It is distributed under the GNU General Public License v2 - see the accompanying COPYING file for more details.

On what hardware does it run?

Although originally developed first for 32-bit x86-based PCs (386 or higher), today Linux also runs on (at least) the Compaq Alpha AXP, Sun SPARC and UltraSPARC, Motorola 68000, PowerPC, PowerPC64, ARM, Hitachi SuperH, Cell, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, DEC VAX, AMD x86-64, AXIS CRIS, Xtensa, Tilera TILE, ARC and Renesas M32R architectures.

Linux is easily portable to most general-purpose 32- or 64-bit architectures as long as they have a paged memory management unit (PMMU) and a port of the GNU C compiler (gcc) (part of The GNU Compiler Collection, GCC). Linux has also been ported to a number of architectures without a PMMU, although functionality is then obviously somewhat limited. Linux has also been ported to itself. You can now run the kernel as a userspace application - this is called UserMode Linux (UML).

Documentation

- There is a lot of documentation available both in electronic form on the Internet and in books, both Linux-specific and pertaining to general UNIX questions. I'd recommend looking into the documentation subdirectories on any Linux FTP site for the LDP (Linux Documentation Project) books. This README is not meant to be documentation on the system: there are much better sources available.
- There are various README files in the Documentation/ subdirectory: these typically contain kernel-specific installation notes for some drivers for example. See Documentation/00-INDEX for a list of what is contained in each file. Please read the Documentation/process/changes.rst file, as it contains information about the problems, which may result by upgrading your kernel.

Installing the kernel source

- If you install the full sources, put the kernel tarball in a directory where you have permissions (e.g. your home directory) and unpack it:

```
xz -cd linux-4.X.tar.xz | tar xvf -
```

  Replace “X” with the version number of the latest kernel.

  Do NOT use the /usr/src/linux area! This area has a (usually incomplete) set of kernel headers that are used by the library header files. They should match the library, and not get messed up by whatever the kernel-du-jour happens to be.

- You can also upgrade between 4.x releases by patching. Patches are distributed in the xz format. To install by patching, get all the newer patch files, enter the top level directory of the kernel source (linux-4.X) and execute:

```
xz -cd ../patch-4.x.xz | patch -p1
```

  Replace “x” for all versions bigger than the version “X” of your current source tree, **in_order**, and you should be ok. You may want to remove the backup files (some-file-name~ or some-file-name.orig), and make sure that there are no failed patches (some-file-name# or some-file-name.rej). If there are, either you or I have made a mistake.

  Unlike patches for the 4.x kernels, patches for the 4.x.y kernels (also known as the -stable kernels) are not incremental but instead apply directly to the base 4.x kernel. For example, if your base kernel is 4.0 and you want to apply the 4.0.3 patch, you must not first apply the 4.0.1 and 4.0.2 patches. Similarly, if you are running kernel version 4.0.2 and want to jump to 4.0.3, you must first reverse the 4.0.2 patch (that is, patch -R) **before** applying the 4.0.3 patch. You can read more on this in Documentation/process/applying-patches.rst .

  Alternatively, the script patch-kernel can be used to automate this process. It determines the current kernel version and applies any patches found:

```
linux/scripts/patch-kernel linux
```

  The first argument in the command above is the location of the kernel source. Patches are applied from the current directory, but an alternative directory can be specified as the second argument.

- Make sure you have no stale .o files and dependencies lying around:

```
cd linux
make mrproper
```

  You should now have the sources correctly installed.

Software requirements

Compiling and running the 4.x kernels requires up-to-date versions of various software packages. Consult Documentation/process/changes.rst for the minimum version numbers required and how to get updates for these packages. Beware that using excessively old versions of these packages can cause indirect errors that are very difficult to track down, so don’t assume that you can just update packages when obvious problems arise during build or operation.

Build directory for the kernel

When compiling the kernel, all output files will per default be stored together with the kernel source code. Using the option `make O=output/dir` allows you to specify an alternate place for

the output files (including .config). Example:

```
kernel source code: /usr/src/linux-4.X
build directory:    /home/name/build/kernel
```

To configure and build the kernel, use:

```
cd /usr/src/linux-4.X
make O=/home/name/build/kernel menuconfig
make O=/home/name/build/kernel
sudo make O=/home/name/build/kernel modules_install install
```

Please note: If the `O=output/dir` option is used, then it must be used for all invocations of make.

Configuring the kernel

Do not skip this step even if you are only upgrading one minor version. New configuration options are added in each release, and odd problems will turn up if the configuration files are not set up as expected. If you want to carry your existing configuration to a new version with minimal work, use `make oldconfig`, which will only ask you for the answers to new questions.

- Alternative configuration commands are:

```
"make config"      Plain text interface.

"make menuconfig"  Text based color menus, radiolists & dialogs.

"make nconfig"     Enhanced text based color menus.

"make xconfig"     Qt based configuration tool.

"make gconfig"     GTK+ based configuration tool.

"make oldconfig"   Default all questions based on the contents of
                   your existing ./.config file and asking about
                   new config symbols.

"make silentoldconfig"
                   Like above, but avoids cluttering the screen
                   with questions already answered.
                   Additionally updates the dependencies.

"make olddefconfig"
                   Like above, but sets new symbols to their default
                   values without prompting.

"make defconfig"   Create a ./.config file by using the default
                   symbol values from either arch/$ARCH/defconfig
                   or arch/$ARCH/configs/${PLATFORM}_defconfig,
                   depending on the architecture.

"make ${PLATFORM}_defconfig"
                   Create a ./.config file by using the default
                   symbol values from
                   arch/$ARCH/configs/${PLATFORM}_defconfig.
                   Use "make help" to get a list of all available
                   platforms of your architecture.

"make allyesconfig"
```

```
                  Create a ./.config file by setting symbol
                  values to 'y' as much as possible.

"make allmodconfig"
                  Create a ./.config file by setting symbol
                  values to 'm' as much as possible.

"make allnoconfig" Create a ./.config file by setting symbol
                  values to 'n' as much as possible.

"make randconfig"  Create a ./.config file by setting symbol
                  values to random values.

"make localmodconfig" Create a config based on current config and
                      loaded modules (lsmod). Disables any module
                      option that is not needed for the loaded modules.

                      To create a localmodconfig for another machine,
                      store the lsmod of that machine into a file
                      and pass it in as a LSMOD parameter.

               target$ lsmod > /tmp/mylsmod
               target$ scp /tmp/mylsmod host:/tmp

               host$ make LSMOD=/tmp/mylsmod localmodconfig

                      The above also works when cross compiling.

"make localyesconfig" Similar to localmodconfig, except it will convert
                      all module options to built in (=y) options.
```

You can find more information on using the Linux kernel config tools in Documentation/kbuild/kconfig.txt.

- NOTES on `make config`:
  - Having unnecessary drivers will make the kernel bigger, and can under some circumstances lead to problems: probing for a nonexistent controller card may confuse your other controllers.
  - A kernel with math-emulation compiled in will still use the coprocessor if one is present: the math emulation will just never get used in that case. The kernel will be slightly larger, but will work on different machines regardless of whether they have a math coprocessor or not.
  - The "kernel hacking" configuration details usually result in a bigger or slower kernel (or both), and can even make the kernel less stable by configuring some routines to actively try to break bad code to find kernel problems (kmalloc()). Thus you should probably answer 'n' to the questions for "development", "experimental", or "debugging" features.

Compiling the kernel

- Make sure you have at least gcc 3.2 available. For more information, refer to Documentation/process/changes.rst .

  Please note that you can still run a.out user programs with this kernel.

- Do a `make` to create a compressed kernel image. It is also possible to do `make install` if you have lilo installed to suit the kernel makefiles, but you may want to check your particular lilo setup first.

  To do the actual install, you have to be root, but none of the normal build should require that. Don't take the name of root in vain.

- If you configured any of the parts of the kernel as `modules`, you will also have to do `make modules_install`.
- Verbose kernel compile/build output:

  Normally, the kernel build system runs in a fairly quiet mode (but not totally silent). However, sometimes you or other kernel developers need to see compile, link, or other commands exactly as they are executed. For this, use "verbose" build mode. This is done by passing `V=1` to the `make` command, e.g.:

  ```
  make V=1 all
  ```

  To have the build system also tell the reason for the rebuild of each target, use `V=2`. The default is `V=0`.
- Keep a backup kernel handy in case something goes wrong. This is especially true for the development releases, since each new release contains new code which has not been debugged. Make sure you keep a backup of the modules corresponding to that kernel, as well. If you are installing a new kernel with the same version number as your working kernel, make a backup of your modules directory before you do a `make modules_install`.

  Alternatively, before compiling, use the kernel config option "LOCALVERSION" to append a unique suffix to the regular kernel version. LOCALVERSION can be set in the "General Setup" menu.
- In order to boot your new kernel, you'll need to copy the kernel image (e.g. .../linux/arch/x86/boot/bzImage after compilation) to the place where your regular bootable kernel is found.
- Booting a kernel directly from a floppy without the assistance of a bootloader such as LILO, is no longer supported.

  If you boot Linux from the hard drive, chances are you use LILO, which uses the kernel image as specified in the file /etc/lilo.conf. The kernel image file is usually /vmlinuz, /boot/vmlinuz, /bzImage or /boot/bzImage. To use the new kernel, save a copy of the old image and copy the new image over the old one. Then, you MUST RERUN LILO to update the loading map! If you don't, you won't be able to boot the new kernel image.

  Reinstalling LILO is usually a matter of running /sbin/lilo. You may wish to edit /etc/lilo.conf to specify an entry for your old kernel image (say, /vmlinux.old) in case the new one does not work. See the LILO docs for more information.

  After reinstalling LILO, you should be all set. Shutdown the system, reboot, and enjoy!

  If you ever need to change the default root device, video mode, ramdisk size, etc. in the kernel image, use the rdev program (or alternatively the LILO boot options when appropriate). No need to recompile the kernel to change these parameters.
- Reboot with the new kernel and enjoy.

If something goes wrong

- If you have problems that seem to be due to kernel bugs, please check the file MAINTAINERS to see if there is a particular person associated with the part of the kernel that you are having trouble with. If there isn't anyone listed there, then the second best thing is to mail them to me (torvalds@linux-foundation.org), and possibly to any other relevant mailing-list or to the newsgroup.
- In all bug-reports, *please* tell what kernel you are talking about, how to duplicate the problem, and what your setup is (use your common sense). If the problem is new, tell me so, and if the problem is old, please try to tell me when you first noticed it.
- If the bug results in a message like:

```
unable to handle kernel paging request at address C0000010
Oops: 0002
EIP:   0010:XXXXXXXX
eax: xxxxxxxx   ebx: xxxxxxxx   ecx: xxxxxxxx   edx: xxxxxxxx
esi: xxxxxxxx   edi: xxxxxxxx   ebp: xxxxxxxx
ds: xxxx  es: xxxx  fs: xxxx  gs: xxxx
Pid: xx, process nr: xx
xx xx xx xx xx xx xx xx xx xx
```

or similar kernel debugging information on your screen or in your system log, please duplicate it *exactly*. The dump may look incomprehensible to you, but it does contain information that may help debugging the problem. The text above the dump is also important: it tells something about why the kernel dumped code (in the above example, it's due to a bad kernel pointer). More information on making sense of the dump is in Documentation/admin-guide/oops-tracing.rst

- If you compiled the kernel with CONFIG_KALLSYMS you can send the dump as is, otherwise you will have to use the `ksymoops` program to make sense of the dump (but compiling with CONFIG_KALLSYMS is usually preferred). This utility can be downloaded from https://www.kernel.org/pub/linux/utils/kernel/ksymoops/ . Alternatively, you can do the dump lookup by hand:
- In debugging dumps like the above, it helps enormously if you can look up what the EIP value means. The hex value as such doesn't help me or anybody else very much: it will depend on your particular kernel setup. What you should do is take the hex value from the EIP line (ignore the `0010:`), and look it up in the kernel namelist to see which kernel function contains the offending address.

  To find out the kernel function name, you'll need to find the system binary associated with the kernel that exhibited the symptom. This is the file 'linux/vmlinux'. To extract the namelist and match it against the EIP from the kernel crash, do:

```
nm vmlinux | sort | less
```

  This will give you a list of kernel addresses sorted in ascending order, from which it is simple to find the function that contains the offending address. Note that the address given by the kernel debugging messages will not necessarily match exactly with the function addresses (in fact, that is very unlikely), so you can't just 'grep' the list: the list will, however, give you the starting point of each kernel function, so by looking for the function that has a starting address lower than the one you are searching for but is followed by a function with a higher address you will find the one you want. In fact, it may be a good idea to include a bit of "context" in your problem report, giving a few lines around the interesting one.

  If you for some reason cannot do the above (you have a pre-compiled kernel image or similar), telling me as much about your setup as possible will help. Please read the *admin-guide/reporting-bugs.rst* document for details.

- Alternatively, you can use gdb on a running kernel. (read-only; i.e. you cannot change values or set break points.) To do this, first compile the kernel with -g; edit arch/x86/Makefile appropriately, then do a `make clean`. You'll also need to enable CONFIG_PROC_FS (via `make config`).

  After you've rebooted with the new kernel, do `gdb vmlinux /proc/kcore`. You can now use all the usual gdb commands. The command to look up the point where your system crashed is `l *0xXXXXXXXX`. (Replace the XXXes with the EIP value.)

  gdb'ing a non-running kernel currently fails because gdb (wrongly) disregards the starting offset for which the kernel is compiled.

CHAPTER

TWO

THE KERNEL'S COMMAND-LINE PARAMETERS

The following is a consolidated list of the kernel parameters as implemented by the __setup(), core_param() and module_param() macros and sorted into English Dictionary order (defined as ignoring all punctuation and sorting digits before letters in a case insensitive manner), and with descriptions where known.

The kernel parses parameters from the kernel command line up to "–"; if it doesn't recognize a parameter and it doesn't contain a '.', the parameter gets passed to init: parameters with '=' go into init's environment, others are passed as command line arguments to init. Everything after "–" is passed as an argument to init.

Module parameters can be specified in two ways: via the kernel command line with a module name prefix, or via modprobe, e.g.:

```
(kernel command line) usbcore.blinkenlights=1
(modprobe command line) modprobe usbcore blinkenlights=1
```

Parameters for modules which are built into the kernel need to be specified on the kernel command line. modprobe looks through the kernel command line (/proc/cmdline) and collects module parameters when it loads a module, so the kernel command line can be used for loadable modules too.

Hyphens (dashes) and underscores are equivalent in parameter names, so:

```
log_buf_len=1M print-fatal-signals=1
```

can also be entered as:

```
log-buf-len=1M print_fatal_signals=1
```

Double-quotes can be used to protect spaces in values, e.g.:

```
param="spaces in here"
```

cpu lists:

Some kernel parameters take a list of CPUs as a value, e.g. isolcpus, nohz_full, irqaffinity, rcu_nocbs. The format of this list is:

<cpu number>,...,<cpu number>

or

<cpu number>-<cpu number> (must be a positive range in ascending order)

or a mixture

<cpu number>,...,<cpu number>-<cpu number>

Note that for the special case of a range one can split the range into equal sized groups and for each group use some amount from the beginning of that group:

<cpu number>-cpu number>:<used size>/<group size>

For example one can add to the command line following parameter:

isolcpus=1,2,10-20,100-2000:2/25

where the final item represents CPUs 100,101,125,126,150,151,...

This document may not be entirely up to date and comprehensive. The command "modinfo -p ${modulename}" shows a current list of all parameters of a loadable module. Loadable modules, after being loaded into the running kernel, also reveal their parameters in /sys/module/${modulename}/parameters/. Some of these parameters may be changed at runtime by the command `echo -n ${value} > /sys/module/${modulename}/parameters/${parm}`.

The parameters listed below are only valid if certain kernel build options were enabled and if respective hardware is present. The text in square brackets at the beginning of each description states the restrictions within which a parameter is applicable:

```
ACPI    ACPI support is enabled.
AGP     AGP (Accelerated Graphics Port) is enabled.
ALSA    ALSA sound support is enabled.
APIC    APIC support is enabled.
APM     Advanced Power Management support is enabled.
ARM     ARM architecture is enabled.
AX25    Appropriate AX.25 support is enabled.
BLACKFIN Blackfin architecture is enabled.
CLK     Common clock infrastructure is enabled.
CMA     Contiguous Memory Area support is enabled.
DRM     Direct Rendering Management support is enabled.
DYNAMIC_DEBUG Build in debug messages and enable them at runtime
EDD     BIOS Enhanced Disk Drive Services (EDD) is enabled
EFI     EFI Partitioning (GPT) is enabled
EIDE    EIDE/ATAPI support is enabled.
EVM     Extended Verification Module
FB      The frame buffer device is enabled.
FTRACE  Function tracing enabled.
GCOV    GCOV profiling is enabled.
HW      Appropriate hardware is enabled.
IA-64   IA-64 architecture is enabled.
IMA     Integrity measurement architecture is enabled.
IOSCHED More than one I/O scheduler is enabled.
IP_PNP  IP DHCP, BOOTP, or RARP is enabled.
IPV6    IPv6 support is enabled.
ISAPNP  ISA PnP code is enabled.
ISDN    Appropriate ISDN support is enabled.
JOY     Appropriate joystick support is enabled.
KGDB    Kernel debugger support is enabled.
KVM     Kernel Virtual Machine support is enabled.
LIBATA  Libata driver is enabled
LP      Printer support is enabled.
LOOP    Loopback device support is enabled.
M68k    M68k architecture is enabled.
                These options have more detailed description inside of
                Documentation/m68k/kernel-options.txt.
MDA     MDA console support is enabled.
MIPS    MIPS architecture is enabled.
MOUSE   Appropriate mouse support is enabled.
MSI     Message Signaled Interrupts (PCI).
MTD     MTD (Memory Technology Device) support is enabled.
NET     Appropriate network support is enabled.
NUMA    NUMA support is enabled.
NFS     Appropriate NFS support is enabled.
OSS     OSS sound support is enabled.
PV_OPS  A paravirtualized kernel is enabled.
```

```
PARIDE  The ParIDE (parallel port IDE) subsystem is enabled.
PARISC  The PA-RISC architecture is enabled.
PCI     PCI bus support is enabled.
PCIE    PCI Express support is enabled.
PCMCIA  The PCMCIA subsystem is enabled.
PNP     Plug & Play support is enabled.
PPC     PowerPC architecture is enabled.
PPT     Parallel port support is enabled.
PS2     Appropriate PS/2 support is enabled.
RAM     RAM disk support is enabled.
RDT     Intel Resource Director Technology.
S390    S390 architecture is enabled.
SCSI    Appropriate SCSI support is enabled.
                A lot of drivers have their options described inside
                the Documentation/scsi/ sub-directory.
SECURITY Different security models are enabled.
SELINUX SELinux support is enabled.
APPARMOR AppArmor support is enabled.
SERIAL  Serial support is enabled.
SH      SuperH architecture is enabled.
SMP     The kernel is an SMP kernel.
SPARC   Sparc architecture is enabled.
SWSUSP  Software suspend (hibernation) is enabled.
SUSPEND System suspend states are enabled.
TPM     TPM drivers are enabled.
TS      Appropriate touchscreen support is enabled.
UMS     USB Mass Storage support is enabled.
USB     USB support is enabled.
USBHID  USB Human Interface Device support is enabled.
V4L     Video For Linux support is enabled.
VMMIO   Driver for memory mapped virtio devices is enabled.
VGA     The VGA console has been enabled.
VT      Virtual terminal support is enabled.
WDT     Watchdog support is enabled.
XT      IBM PC/XT MFM hard disk support is enabled.
X86-32  X86-32, aka i386 architecture is enabled.
X86-64  X86-64 architecture is enabled.
                More X86-64 boot options can be found in
                Documentation/x86/x86_64/boot-options.txt .
X86     Either 32-bit or 64-bit x86 (same as X86-32+X86-64)
X86_UV  SGI UV support is enabled.
XEN     Xen support is enabled
```

In addition, the following text indicates that the option:

```
BUGS=   Relates to possible processor bugs on the said processor.
KNL     Is a kernel start-up parameter.
BOOT    Is a boot loader parameter.
```

Parameters denoted with BOOT are actually interpreted by the boot loader, and have no meaning to the kernel directly. Do not modify the syntax of boot loader parameters without extreme need or coordination with <Documentation/x86/boot.txt>.

There are also arch-specific kernel-parameters not documented here. See for example <Documentation/x86/x86_64/boot-options.txt>.

Note that ALL kernel parameters listed below are CASE SENSITIVE, and that a trailing = on the name of any parameter states that that parameter will be entered as an environment variable, whereas its absence indicates that it will appear as a kernel argument readable via /proc/cmdline by programs running once the system is up.

The number of kernel parameters is not limited, but the length of the complete command line (parameters including spaces etc.) is limited to a fixed number of characters. This limit depends on the ar-

chitecture and is between 256 and 4096 characters. It is defined in the file ./include/asm/setup.h as COMMAND_LINE_SIZE.

Finally, the [KMG] suffix is commonly described after a number of kernel parameter values. These 'K', 'M', and 'G' letters represent the _binary_ multipliers 'Kilo', 'Mega', and 'Giga', equaling 2^10, 2^20, and 2^30 bytes respectively. Such letter suffixes can also be entirely omitted:

```
        acpi=           [HW,ACPI,X86,ARM64]
                        Advanced Configuration and Power Interface
                        Format: { force | on | off | strict | noirq | rsdt |
                                  copy_dsdt }
                        force -- enable ACPI if default was off
                        on -- enable ACPI but allow fallback to DT [arm64]
                        off -- disable ACPI if default was on
                        noirq -- do not use ACPI for IRQ routing
                        strict -- Be less tolerant of platforms that are not
                                strictly ACPI specification compliant.
                        rsdt -- prefer RSDT over (default) XSDT
                        copy_dsdt -- copy DSDT to memory
                        For ARM64, ONLY ``acpi=off'', ``acpi=on'' or ``acpi=force''
                        are available

                        See also Documentation/power/runtime_pm.txt, pci=noacpi

        acpi_apic_instance=     [ACPI, IOAPIC]
                        Format: <int>
                        2: use 2nd APIC table, if available
                        1,0: use 1st APIC table
                        default: 0

        acpi_backlight= [HW,ACPI]
                        acpi_backlight=vendor
                        acpi_backlight=video
                        If set to vendor, prefer vendor specific driver
                        (e.g. thinkpad_acpi, sony_acpi, etc.) instead
                        of the ACPI video.ko driver.

        acpi_force_32bit_fadt_addr
                        force FADT to use 32 bit addresses rather than the
                        64 bit X_* addresses. Some firmware have broken 64
                        bit addresses for force ACPI ignore these and use
                        the older legacy 32 bit addresses.

        acpica_no_return_repair [HW, ACPI]
                        Disable AML predefined validation mechanism
                        This mechanism can repair the evaluation result to make
                        the return objects more ACPI specification compliant.
                        This option is useful for developers to identify the
                        root cause of an AML interpreter issue when the issue
                        has something to do with the repair mechanism.

        acpi.debug_layer=       [HW,ACPI,ACPI_DEBUG]
        acpi.debug_level=       [HW,ACPI,ACPI_DEBUG]
                        Format: <int>
                        CONFIG_ACPI_DEBUG must be enabled to produce any ACPI
                        debug output.  Bits in debug_layer correspond to a
                        _COMPONENT in an ACPI source file, e.g.,
                            #define _COMPONENT ACPI_PCI_COMPONENT
                        Bits in debug_level correspond to a level in
```

```
                ACPI_DEBUG_PRINT statements, e.g.,
                    ACPI_DEBUG_PRINT((ACPI_DB_INFO, ...
                The debug_level mask defaults to ``info''.  See
                Documentation/acpi/debug.txt for more information about
                debug layers and levels.

                Enable processor driver info messages:
                    acpi.debug_layer=0x20000000
                Enable PCI/PCI interrupt routing info messages:
                    acpi.debug_layer=0x400000
                Enable AML ``Debug'' output, i.e., stores to the Debug
                object while interpreting AML:
                    acpi.debug_layer=0xffffffff acpi.debug_level=0x2
                Enable all messages related to ACPI hardware:
                    acpi.debug_layer=0x2 acpi.debug_level=0xffffffff

                Some values produce so much output that the system is
                unusable.  The ``log_buf_len'' parameter may be useful
                if you need to capture more output.

        acpi_enforce_resources= [ACPI]
                { strict | lax | no }
                Check for resource conflicts between native drivers
                and ACPI OperationRegions (SystemIO and SystemMemory
                only). IO ports and memory declared in ACPI might be
                used by the ACPI subsystem in arbitrary AML code and
                can interfere with legacy drivers.
                strict (default): access to resources claimed by ACPI
                is denied; legacy drivers trying to access reserved
                resources will fail to bind to device using them.
                lax: access to resources claimed by ACPI is allowed;
                legacy drivers trying to access reserved resources
                will bind successfully but a warning message is logged.
                no: ACPI OperationRegions are not marked as reserved,
                no further checks are performed.

        acpi_force_table_verification   [HW,ACPI]
                Enable table checksum verification during early stage.
                By default, this is disabled due to x86 early mapping
                size limitation.

        acpi_irq_balance [HW,ACPI]
                ACPI will balance active IRQs
                default in APIC mode

        acpi_irq_nobalance [HW,ACPI]
                ACPI will not move active IRQs (default)
                default in PIC mode

        acpi_irq_isa=   [HW,ACPI] If irq_balance, mark listed IRQs used by ISA
                Format: <irq>,<irq>...

        acpi_irq_pci=   [HW,ACPI] If irq_balance, clear listed IRQs for
                use by PCI
                Format: <irq>,<irq>...

        acpi_mask_gpe=  [HW,ACPI]
                Due to the existence of _Lxx/_Exx, some GPEs triggered
```

```
                by unsupported hardware/firmware features can result in
                GPE floodings that cannot be automatically disabled by
                the GPE dispatcher.
                This facility can be used to prevent such uncontrolled
                GPE floodings.
                Format: <int>
                Support masking of GPEs numbered from 0x00 to 0x7f.

acpi_no_auto_serialize  [HW,ACPI]
                Disable auto-serialization of AML methods
                AML control methods that contain the opcodes to create
                named objects will be marked as ``Serialized'' by the
                auto-serialization feature.
                This feature is enabled by default.
                This option allows to turn off the feature.

acpi_no_memhotplug [ACPI] Disable memory hotplug.  Useful for kdump
                   kernels.

acpi_no_static_ssdt     [HW,ACPI]
                Disable installation of static SSDTs at early boot time
                By default, SSDTs contained in the RSDT/XSDT will be
                installed automatically and they will appear under
                /sys/firmware/acpi/tables.
                This option turns off this feature.
                Note that specifying this option does not affect
                dynamic table installation which will install SSDT
                tables to /sys/firmware/acpi/tables/dynamic.

acpi_rsdp=      [ACPI,EFI,KEXEC]
                Pass the RSDP address to the kernel, mostly used
                on machines running EFI runtime service to boot the
                second kernel for kdump.

acpi_os_name=   [HW,ACPI] Tell ACPI BIOS the name of the OS
                Format: To spoof as Windows 98: =''Microsoft Windows''

acpi_rev_override [ACPI] Override the _REV object to return 5 (instead
                of 2 which is mandated by ACPI 6) as the supported ACPI
                specification revision (when using this switch, it may
                be necessary to carry out a cold reboot _twice_ in a
                row to make it take effect on the platform firmware).

acpi_osi=       [HW,ACPI] Modify list of supported OS interface strings
                acpi_osi=''string1''      # add string1
                acpi_osi='''!string2''    # remove string2
                acpi_osi=!*             # remove all strings
                acpi_osi=!              # disable all built-in OS vendor
                                          strings
                acpi_osi=!!             # enable all built-in OS vendor
                                          strings
                acpi_osi=               # disable all strings

                `acpi_osi=!' can be used in combination with single or
                multiple `acpi_osi=''string1''' to support specific OS
                vendor string(s).  Note that such command can only
                affect the default state of the OS vendor strings, thus
                it cannot affect the default state of the feature group
```

```
                    strings and the current state of the OS vendor strings,
                    specifying it multiple times through kernel command line
                    is meaningless.  This command is useful when one do not
                    care about the state of the feature group strings which
                    should be controlled by the OSPM.
                    Examples:
                      1. `acpi_osi=! acpi_osi=''Windows 2000''' is equivalent
                         to `acpi_osi=''Windows 2000'' acpi_osi=!', they all
                         can make `_OSI(``Windows 2000'')' TRUE.

                    `acpi_osi=' cannot be used in combination with other
                    `acpi_osi=' command lines, the _OSI method will not
                    exist in the ACPI namespace.  NOTE that such command can
                    only affect the _OSI support state, thus specifying it
                    multiple times through kernel command line is also
                    meaningless.
                    Examples:
                      1. `acpi_osi=' can make `CondRefOf(_OSI, Local1)'
                         FALSE.

                    `acpi_osi=!*' can be used in combination with single or
                    multiple `acpi_osi=''string1''' to support specific
                    string(s).  Note that such command can affect the
                    current state of both the OS vendor strings and the
                    feature group strings, thus specifying it multiple times
                    through kernel command line is meaningful.  But it may
                    still not able to affect the final state of a string if
                    there are quirks related to this string.  This command
                    is useful when one want to control the state of the
                    feature group strings to debug BIOS issues related to
                    the OSPM features.
                    Examples:
                      1. `acpi_osi=''Module Device'' acpi_osi=!*' can make
                         `_OSI(``Module Device'')' FALSE.
                      2. `acpi_osi=!* acpi_osi=''Module Device''' can make
                         `_OSI(``Module Device'')' TRUE.
                      3. `acpi_osi=! acpi_osi=!* acpi_osi=''Windows 2000''' is
                         equivalent to
                         `acpi_osi=!* acpi_osi=! acpi_osi=''Windows 2000'''
                         and
                         `acpi_osi=!* acpi_osi=''Windows 2000'' acpi_osi=!',
                         they all will make `_OSI(``Windows 2000'')' TRUE.

    acpi_pm_good    [X86]
                    Override the pmtimer bug detection: force the kernel
                    to assume that this machine's pmtimer latches its value
                    and always returns good values.

    acpi_sci=       [HW,ACPI] ACPI System Control Interrupt trigger mode
                    Format: { level | edge | high | low }

    acpi_skip_timer_override [HW,ACPI]
                    Recognize and ignore IRQ0/pin2 Interrupt Override.
                    For broken nForce2 BIOS resulting in XT-PIC timer.

    acpi_sleep=     [HW,ACPI] Sleep options
                    Format: { s3_bios, s3_mode, s3_beep, s4_nohwsig,
                              old_ordering, nonvs, sci_force_enable }
```

```
                See Documentation/power/video.txt for information on
                s3_bios and s3_mode.
                s3_beep is for debugging; it makes the PC's speaker beep
                as soon as the kernel's real-mode entry point is called.
                s4_nohwsig prevents ACPI hardware signature from being
                used during resume from hibernation.
                old_ordering causes the ACPI 1.0 ordering of the _PTS
                control method, with respect to putting devices into
                low power states, to be enforced (the ACPI 2.0 ordering
                of _PTS is used by default).
                nonvs prevents the kernel from saving/restoring the
                ACPI NVS memory during suspend/hibernation and resume.
                sci_force_enable causes the kernel to set SCI_EN directly
                on resume from S1/S3 (which is against the ACPI spec,
                but some broken systems don't work without it).

acpi_use_timer_override [HW,ACPI]
                Use timer override. For some broken Nvidia NF5 boards
                that require a timer override, but don't have HPET

add_efi_memmap  [EFI; X86] Include EFI memory map in
                kernel's map of available physical RAM.

agp=            [AGP]
                { off | try_unsupported }
                off: disable AGP support
                try_unsupported: try to drive unsupported chipsets
                        (may crash computer or cause data corruption)

ALSA            [HW,ALSA]
                See Documentation/sound/alsa/alsa-parameters.txt

alignment=      [KNL,ARM]
                Allow the default userspace alignment fault handler
                behaviour to be specified.  Bit 0 enables warnings,
                bit 1 enables fixups, and bit 2 sends a segfault.

align_va_addr=  [X86-64]
                Align virtual addresses by clearing slice [14:12] when
                allocating a VMA at process creation time. This option
                gives you up to 3% performance improvement on AMD F15h
                machines (where it is enabled by default) for a
                CPU-intensive style benchmark, and it can vary highly in
                a microbenchmark depending on workload and compiler.

                32: only for 32-bit processes
                64: only for 64-bit processes
                on: enable for both 32- and 64-bit processes
                off: disable for both 32- and 64-bit processes

alloc_snapshot  [FTRACE]
                Allocate the ftrace snapshot buffer on boot up when the
                main buffer is allocated. This is handy if debugging
                and you need to use tracing_snapshot() on boot up, and
                do not want to use tracing_snapshot_alloc() as it needs
                to be done where GFP_KERNEL allocations are allowed.

amd_iommu=      [HW,X86-64]
```

```
                Pass parameters to the AMD IOMMU driver in the system.
                Possible values are:
                fullflush - enable flushing of IO/TLB entries when
                            they are unmapped. Otherwise they are
                            flushed before they will be reused, which
                            is a lot of faster
                off       - do not initialize any AMD IOMMU found in
                            the system
                force_isolation - Force device isolation for all
                                  devices. The IOMMU driver is not
                                  allowed anymore to lift isolation
                                  requirements as needed. This option
                                  does not override iommu=pt

amd_iommu_dump= [HW,X86-64]
                Enable AMD IOMMU driver option to dump the ACPI table
                for AMD IOMMU. With this option enabled, AMD IOMMU
                driver will print ACPI tables for AMD IOMMU during
                IOMMU initialization.

amd_iommu_intr= [HW,X86-64]
                Specifies one of the following AMD IOMMU interrupt
                remapping modes:
                legacy     - Use legacy interrupt remapping mode.
                vapic      - Use virtual APIC mode, which allows IOMMU
                             to inject interrupts directly into guest.
                             This mode requires kvm-amd.avic=1.
                             (Default when IOMMU HW support is present.)

amijoy.map=     [HW,JOY] Amiga joystick support
                Map of devices attached to JOY0DAT and JOY1DAT
                Format: <a>,<b>
                See also Documentation/input/joystick.txt

analog.map=     [HW,JOY] Analog joystick and gamepad support
                Specifies type or capabilities of an analog joystick
                connected to one of 16 gameports
                Format: <type1>,<type2>,..<type16>

apc=            [HW,SPARC]
                Power management functions (SPARCstation-4/5 + deriv.)
                Format: noidle
                Disable APC CPU standby support. SPARCstation-Fox does
                not play well with APC CPU idle - disable it if you have
                APC and your system crashes randomly.

apic=           [APIC,X86-32] Advanced Programmable Interrupt Controller
                Change the output verbosity whilst booting
                Format: { quiet (default) | verbose | debug }
                Change the amount of debugging information output
                when initialising the APIC and IO-APIC components.

apic_extnmi=    [APIC,X86] External NMI delivery setting
                Format: { bsp (default) | all | none }
                bsp:  External NMI is delivered only to CPU 0
                all:  External NMIs are broadcast to all CPUs as a
                      backup of CPU 0
                none: External NMI is masked for all CPUs. This is
```

```
                        useful so that a dump capture kernel won't be
                        shot down by NMI

    autoconf=       [IPV6]
                    See Documentation/networking/ipv6.txt.

    show_lapic=     [APIC,X86] Advanced Programmable Interrupt Controller
                    Limit apic dumping. The parameter defines the maximal
                    number of local apics being dumped. Also it is possible
                    to set it to ``all'' by meaning -- no limit here.
                    Format: { 1 (default) | 2 | ... | all }.
                    The parameter valid if only apic=debug or
                    apic=verbose is specified.
                    Example: apic=debug show_lapic=all

    apm=            [APM] Advanced Power Management
                    See header of arch/x86/kernel/apm_32.c.

    arcrimi=        [HW,NET] ARCnet - ``RIM I'' (entirely mem-mapped) cards
                    Format: <io>,<irq>,<nodeID>

    ataflop=        [HW,M68k]

    atarimouse=     [HW,MOUSE] Atari Mouse

    atkbd.extra=    [HW] Enable extra LEDs and keys on IBM RapidAccess,
                    EzKey and similar keyboards

    atkbd.reset=    [HW] Reset keyboard during initialization

    atkbd.set=      [HW] Select keyboard code set
                    Format: <int> (2 = AT (default), 3 = PS/2)

    atkbd.scroll=   [HW] Enable scroll wheel on MS Office and similar
                    keyboards

    atkbd.softraw=  [HW] Choose between synthetic and real raw mode
                    Format: <bool> (0 = real, 1 = synthetic (default))

    atkbd.softrepeat= [HW]
                    Use software keyboard repeat

    audit=          [KNL] Enable the audit sub-system
                    Format: { ``0'' | ``1'' } (0 = disabled, 1 = enabled)
                    0 - kernel audit is disabled and can not be enabled
                        until the next reboot
                    unset - kernel audit is initialized but disabled and
                        will be fully enabled by the userspace auditd.
                    1 - kernel audit is initialized and partially enabled,
                        storing at most audit_backlog_limit messages in
                        RAM until it is fully enabled by the userspace
                        auditd.
                    Default: unset

    audit_backlog_limit= [KNL] Set the audit queue size limit.
                    Format: <int> (must be >=0)
                    Default: 64
```

```
bau=            [X86_UV] Enable the BAU on SGI UV.  The default
                behavior is to disable the BAU (i.e. bau=0).
                Format: { ``0'' | ``1'' }
                0 - Disable the BAU.
                1 - Enable the BAU.
                unset - Disable the BAU.

baycom_epp=     [HW,AX25]
                Format: <io>,<mode>

baycom_par=     [HW,AX25] BayCom Parallel Port AX.25 Modem
                Format: <io>,<mode>
                See header of drivers/net/hamradio/baycom_par.c.

baycom_ser_fdx= [HW,AX25]
                BayCom Serial Port AX.25 Modem (Full Duplex Mode)
                Format: <io>,<irq>,<mode>[,<baud>]
                See header of drivers/net/hamradio/baycom_ser_fdx.c.

baycom_ser_hdx= [HW,AX25]
                BayCom Serial Port AX.25 Modem (Half Duplex Mode)
                Format: <io>,<irq>,<mode>
                See header of drivers/net/hamradio/baycom_ser_hdx.c.

blkdevparts=    Manual partition parsing of block device(s) for
                embedded devices based on command line input.
                See Documentation/block/cmdline-partition.txt

boot_delay=     Milliseconds to delay each printk during boot.
                Values larger than 10 seconds (10000) are changed to
                no delay (0).
                Format: integer

bootmem_debug   [KNL] Enable bootmem allocator debug messages.

bert_disable    [ACPI]
                Disable BERT OS support on buggy BIOSes.

bttv.card=      [HW,V4L] bttv (bt848 + bt878 based grabber cards)
bttv.radio=     Most important insmod options are available as
                kernel args too.
bttv.pll=       See Documentation/video4linux/bttv/Insmod-options
bttv.tuner=

bulk_remove=off [PPC]  This parameter disables the use of the pSeries
                firmware feature for flushing multiple hpte entries
                at a time.

c101=           [NET] Moxa C101 synchronous serial card

cachesize=      [BUGS=X86-32] Override level 2 CPU cache size detection.
                Sometimes CPU hardware bugs make them report the cache
                size incorrectly. The kernel will attempt work arounds
                to fix known problems, but for some CPUs it is not
                possible to determine what the correct size should be.
                This option provides an override for these situations.

ca_keys=        [KEYS] This parameter identifies a specific key(s) on
```

```
                the system trusted keyring to be used for certificate
                trust validation.
                format: { id:<keyid> | builtin }

cca=            [MIPS] Override the kernel pages' cache coherency
                algorithm.  Accepted values range from 0 to 7
                inclusive. See arch/mips/include/asm/pgtable-bits.h
                for platform specific values (SB1, Loongson3 and
                others).

ccw_timeout_log [S390]
                See Documentation/s390/CommonIO for details.

cgroup_disable= [KNL] Disable a particular controller
                Format: {name of the controller(s) to disable}
                The effects of cgroup_disable=foo are:
                - foo isn't auto-mounted if you mount all cgroups in
                  a single hierarchy
                - foo isn't visible as an individually mountable
                  subsystem
                {Currently only ``memory'' controller deal with this and
                cut the overhead, others just disable the usage. So
                only cgroup_disable=memory is actually worthy}

cgroup_no_v1=   [KNL] Disable one, multiple, all cgroup controllers in v1
                Format: { controller[,controller...] | ``all'' }
                Like cgroup_disable, but only applies to cgroup v1;
                the blacklisted controllers remain available in cgroup2.

cgroup.memory=  [KNL] Pass options to the cgroup memory controller.
                Format: <string>
                nosocket -- Disable socket memory accounting.
                nokmem -- Disable kernel memory accounting.

checkreqprot    [SELINUX] Set initial checkreqprot flag value.
                Format: { ``0'' | ``1'' }
                See security/selinux/Kconfig help text.
                0 -- check protection applied by kernel (includes
                        any implied execute protection).
                1 -- check protection requested by application.
                Default value is set via a kernel config option.
                Value can be changed at runtime via
                        /selinux/checkreqprot.

cio_ignore=     [S390]
                See Documentation/s390/CommonIO for details.
clk_ignore_unused
                [CLK]
                Prevents the clock framework from automatically gating
                clocks that have not been explicitly enabled by a Linux
                device driver but are enabled in hardware at reset or
                by the bootloader/firmware. Note that this does not
                force such clocks to be always-on nor does it reserve
                those clocks in any way. This parameter is useful for
                debug and development, but should not be needed on a
                platform with proper driver support.  For more
                information, see Documentation/clk.txt.
```

```
clock=          [BUGS=X86-32, HW] gettimeofday clocksource override.
                [Deprecated]
                Forces specified clocksource (if available) to be used
                when calculating gettimeofday(). If specified
                clocksource is not available, it defaults to PIT.
                Format: { pit | tsc | cyclone | pmtmr }

clocksource=    Override the default clocksource
                Format: <string>
                Override the default clocksource and use the clocksource
                with the name specified.
                Some clocksource names to choose from, depending on
                the platform:
                [all] jiffies (this is the base, fallback clocksource)
                [ACPI] acpi_pm
                [ARM] imx_timer1,OSTS,netx_timer,mpu_timer2,
                        pxa_timer,timer3,32k_counter,timer0_1
                [X86-32] pit,hpet,tsc;
                        scx200_hrt on Geode; cyclone on IBM x440
                [MIPS] MIPS
                [PARISC] cr16
                [S390] tod
                [SH] SuperH
                [SPARC64] tick
                [X86-64] hpet,tsc

clocksource.arm_arch_timer.evtstrm=
                [ARM,ARM64]
                Format: <bool>
                Enable/disable the eventstream feature of the ARM
                architected timer so that code using WFE-based polling
                loops can be debugged more effectively on production
                systems.

clearcpuid=BITNUM [X86]
                Disable CPUID feature X for the kernel. See
                arch/x86/include/asm/cpufeatures.h for the valid bit
                numbers. Note the Linux specific bits are not necessarily
                stable over kernel options, but the vendor specific
                ones should be.
                Also note that user programs calling CPUID directly
                or using the feature without checking anything
                will still see it. This just prevents it from
                being used by the kernel or shown in /proc/cpuinfo.
                Also note the kernel might malfunction if you disable
                some critical bits.

cma=nn[MG]@[start[MG][-end[MG]]]
                [ARM,X86,KNL]
                Sets the size of kernel global memory area for
                contiguous memory allocations and optionally the
                placement constraint by the physical address range of
                memory allocations. A value of 0 disables CMA
                altogether. For more information, see
                include/linux/dma-contiguous.h

cmo_free_hint=  [PPC] Format: { yes | no }
                Specify whether pages are marked as being inactive
```

```
                when they are freed.  This is used in CMO environments
                to determine OS memory pressure for page stealing by
                a hypervisor.
                Default: yes

coherent_pool=nn[KMG]   [ARM,KNL]
                Sets the size of memory pool for coherent, atomic dma
                allocations, by default set to 256K.

code_bytes      [X86] How many bytes of object code to print
                in an oops report.
                Range: 0 - 8192
                Default: 64

com20020=       [HW,NET] ARCnet - COM20020 chipset
                Format:
                <io>[,<irq>[,<nodeID>[,<backplane>[,<ckp>[,<timeout>]]]]]

com90io=        [HW,NET] ARCnet - COM90xx chipset (IO-mapped buffers)
                Format: <io>[,<irq>]

com90xx=        [HW,NET]
                ARCnet - COM90xx chipset (memory-mapped buffers)
                Format: <io>[,<irq>[,<memstart>]]

condev=         [HW,S390] console device
conmode=

console=        [KNL] Output console device and 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
                omit it).  Default is ``9600n8''.

                See Documentation/admin-guide/serial-console.rst for more
                information.  See
                Documentation/networking/netconsole.txt for an
                alternative.

        uart[8250],io,<addr>[,options]
        uart[8250],mmio,<addr>[,options]
        uart[8250],mmio16,<addr>[,options]
        uart[8250],mmio32,<addr>[,options]
        uart[8250],0x<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 matching ttyS device later.
                MMIO inter-register address stride is either 8-bit
                (mmio), 16-bit (mmio16), or 32-bit (mmio32).
                If none of [io|mmio|mmio16|mmio32], <addr> is assumed
                to be equivalent to `mmio'. `options' are specified in
                the same format described for ttyS above; if unspecified,
```

```
                the h/w is not re-initialized.

        hvc<n>  Use the hypervisor console device <n>. This is for
                both Xen and PowerPC hypervisors.

        If the device connected to the port is not a TTY but a braille
        device, prepend ``brl,'' before the device type, for instance
                console=brl,ttyS0
        For now, only VisioBraille is supported.

consoleblank=   [KNL] The console blank (screen saver) timeout in
                seconds. Defaults to 10*60 = 10mins. A value of 0
                disables the blank timer.

coredump_filter=
                [KNL] Change the default value for
                /proc/<pid>/coredump_filter.
                See also Documentation/filesystems/proc.txt.

coresight_cpu_debug.enable
                [ARM,ARM64]
                Format: <bool>
                Enable/disable the CPU sampling based debugging.
                0: default value, disable debugging
                1: enable debugging at boot time

cpuidle.off=1   [CPU_IDLE]
                disable the cpuidle sub-system

cpufreq.off=1   [CPU_FREQ]
                disable the cpufreq sub-system

cpu_init_udelay=N
                [X86] Delay for N microsec between assert and de-assert
                of APIC INIT to start processors.  This delay occurs
                on every CPU online, such as boot, and resume from suspend.
                Default: 10000

cpcihp_generic= [HW,PCI] Generic port I/O CompactPCI driver
                Format:
                <first_slot>,<last_slot>,<port>,<enum_bit>[,<debug>]

crashkernel=size[KMG][@offset[KMG]]
                [KNL] Using kexec, Linux can switch to a `crash kernel'
                upon panic. This parameter reserves the physical
                memory region [offset, offset + size] for that kernel
                image. If `@offset' is omitted, then a suitable offset
                is selected automatically. Check
                Documentation/kdump/kdump.txt for further details.

crashkernel=range1:size1[,range2:size2,...][@offset]
                [KNL] Same as above, but depends on the memory
                in the running system. The syntax of range is
                start-[end] where start and end are both
                a memory unit (amount[KMG]). See also
                Documentation/kdump/kdump.txt for an example.

crashkernel=size[KMG],high
```

```
                [KNL, x86_64] range could be above 4G. Allow kernel
                to allocate physical memory region from top, so could
                be above 4G if system have more than 4G ram installed.
                Otherwise memory region will be allocated below 4G, if
                available.
                It will be ignored if crashkernel=X is specified.
crashkernel=size[KMG],low
                [KNL, x86_64] range under 4G. When crashkernel=X,high
                is passed, kernel could allocate physical memory region
                above 4G, that cause second kernel crash on system
                that require some amount of low memory, e.g. swiotlb
                requires at least 64M+32K low memory, also enough extra
                low memory is needed to make sure DMA buffers for 32-bit
                devices won't run out. Kernel would try to allocate at
                at least 256M below 4G automatically.
                This one let user to specify own low range under 4G
                for second kernel instead.
                0: to disable low allocation.
                It will be ignored when crashkernel=X,high is not used
                or memory reserved is below 4G.

cryptomgr.notests
                [KNL] Disable crypto self-tests

cs89x0_dma=     [HW,NET]
                Format: <dma>

cs89x0_media=   [HW,NET]
                Format: { rj45 | aui | bnc }

dasd=           [HW,NET]
                See header of drivers/s390/block/dasd_devmap.c.

db9.dev[2|3]=   [HW,JOY] Multisystem joystick support via parallel port
                (one device per port)
                Format: <port#>,<type>
                See also Documentation/input/joystick-parport.txt

ddebug_query=   [KNL,DYNAMIC_DEBUG] Enable debug messages at early boot
                time. See
                Documentation/admin-guide/dynamic-debug-howto.rst for
                details.  Deprecated, see dyndbg.

debug           [KNL] Enable kernel debugging (events log level).

debug_locks_verbose=
                [KNL] verbose self-tests
                Format=<0|1>
                Print debugging info while doing the locking API
                self-tests.
                We default to 0 (no extra messages), setting it to
                1 will print _a lot_ more information - normally
                only useful to kernel developers.

debug_objects   [KNL] Enable object debugging

no_debug_objects
                [KNL] Disable object debugging
```

```
debug_guardpage_minorder=
                [KNL] When CONFIG_DEBUG_PAGEALLOC is set, this
                parameter allows control of the order of pages that will
                be intentionally kept free (and hence protected) by the
                buddy allocator. Bigger value increase the probability
                of catching random memory corruption, but reduce the
                amount of memory for normal system use. The maximum
                possible value is MAX_ORDER/2.  Setting this parameter
                to 1 or 2 should be enough to identify most random
                memory corruption problems caused by bugs in kernel or
                driver code when a CPU writes to (or reads from) a
                random memory location. Note that there exists a class
                of memory corruptions problems caused by buggy H/W or
                F/W or by drivers badly programing DMA (basically when
                memory is written at bus level and the CPU MMU is
                bypassed) which are not detectable by
                CONFIG_DEBUG_PAGEALLOC, hence this option will not help
                tracking down these problems.

debug_pagealloc=
                [KNL] When CONFIG_DEBUG_PAGEALLOC is set, this
                parameter enables the feature at boot time. In
                default, it is disabled. We can avoid allocating huge
                chunk of memory for debug pagealloc if we don't enable
                it at boot time and the system will work mostly same
                with the kernel built without CONFIG_DEBUG_PAGEALLOC.
                on: enable the feature

debugpat        [X86] Enable PAT debugging

decnet.addr=    [HW,NET]
                Format: <area>[,<node>]
                See also Documentation/networking/decnet.txt.

default_hugepagesz=
                [same as hugepagesz=] The size of the default
                HugeTLB page size. This is the size represented by
                the legacy /proc/ hugepages APIs, used for SHM, and
                default size when mounting hugetlbfs filesystems.
                Defaults to the default architecture's huge page size
                if not specified.

dhash_entries=  [KNL]
                Set number of hash buckets for dentry cache.

disable_1tb_segments [PPC]
                Disables the use of 1TB hash page table segments. This
                causes the kernel to fall back to 256MB segments which
                can be useful when debugging issues that require an SLB
                miss to occur.

disable=        [IPV6]
                See Documentation/networking/ipv6.txt.

disable_radix   [PPC]
                Disable RADIX MMU mode on POWER9
```

```
disable_cpu_apicid= [X86,APIC,SMP]
                Format: <int>
                The number of initial APIC ID for the
                corresponding CPU to be disabled at boot,
                mostly used for the kdump 2nd kernel to
                disable BSP to wake up multiple CPUs without
                causing system reset or hang due to sending
                INIT from AP to BSP.

disable_ddw     [PPC/PSERIES]
                Disable Dynamic DMA Window support. Use this if
                to workaround buggy firmware.

disable_ipv6=   [IPV6]
                See Documentation/networking/ipv6.txt.

disable_mtrr_cleanup [X86]
                The kernel tries to adjust MTRR layout from continuous
                to discrete, to make X server driver able to add WB
                entry later. This parameter disables that.

disable_mtrr_trim [X86, Intel and AMD only]
                By default the kernel will trim any uncacheable
                memory out of your available memory pool based on
                MTRR settings.  This parameter disables that behavior,
                possibly causing your machine to run very slowly.

disable_timer_pin_1 [X86]
                Disable PIN 1 of APIC timer
                Can be useful to work around chipset bugs.

dis_ucode_ldr   [X86] Disable the microcode loader.

dma_debug=off   If the kernel is compiled with DMA_API_DEBUG support,
                this option disables the debugging code at boot.

dma_debug_entries=<number>
                This option allows to tune the number of preallocated
                entries for DMA-API debugging code. One entry is
                required per DMA-API allocation. Use this if the
                DMA-API debugging code disables itself because the
                architectural default is too low.

dma_debug_driver=<driver_name>
                With this option the DMA-API debugging driver
                filter feature can be enabled at boot time. Just
                pass the driver to filter for as the parameter.
                The filter can be disabled or changed to another
                driver later using sysfs.

drm_kms_helper.edid_firmware=[<connector>:]<file>[,[<connector>:]<file>]
                Broken monitors, graphic adapters, KVMs and EDIDless
                panels may send no or incorrect EDID data sets.
                This parameter allows to specify an EDID data sets
                in the /lib/firmware directory that are used instead.
                Generic built-in EDID data sets are used, if one of
                edid/1024x768.bin, edid/1280x1024.bin,
                edid/1680x1050.bin, or edid/1920x1080.bin is given
```

```
                and no file with the same name exists. Details and
                instructions how to build your own EDID data are
                available in Documentation/EDID/HOWTO.txt. An EDID
                data set will only be used for a particular connector,
                if its name and a colon are prepended to the EDID
                name. Each connector may use a unique EDID data
                set by separating the files with a comma.  An EDID
                data set with no connector name will be used for
                any connectors not explicitly specified.

dscc4.setup=    [NET]

dt_cpu_ftrs=    [PPC]
                Format: {``off'' | ``known''}
                Control how the dt_cpu_ftrs device-tree binding is
                used for CPU feature discovery and setup (if it
                exists).
                off: Do not use it, fall back to legacy cpu table.
                known: Do not pass through unknown features to guests
                or userspace, only those that the kernel is aware of.

dump_apple_properties   [X86]
                Dump name and content of EFI device properties on
                x86 Macs.  Useful for driver authors to determine
                what data is available or for reverse-engineering.

dyndbg[=''val'']          [KNL,DYNAMIC_DEBUG]
module.dyndbg[=''val'']
                Enable debug messages at boot time.  See
                Documentation/admin-guide/dynamic-debug-howto.rst
                for details.

nompx           [X86] Disables Intel Memory Protection Extensions.
                See Documentation/x86/intel_mpx.txt for more
                information about the feature.

nopku           [X86] Disable Memory Protection Keys CPU feature found
                in some Intel CPUs.

module.async_probe [KNL]
                Enable asynchronous probe on this module.

early_ioremap_debug [KNL]
                Enable debug messages in early_ioremap support. This
                is useful for tracking down temporary early mappings
                which are not unmapped.

earlycon=       [KNL] Output early console device and options.

                When used with no options, the early console is
                determined by the stdout-path property in device
                tree's chosen node.

        cdns,<addr>[,options]
                Start an early, polled-mode console on a Cadence
                (xuartps) serial port at the specified address. Only
                supported option is baud rate. If baud rate is not
                specified, the serial port must already be setup and
```

```
        configured.

uart[8250],io,<addr>[,options]
uart[8250],mmio,<addr>[,options]
uart[8250],mmio32,<addr>[,options]
uart[8250],mmio32be,<addr>[,options]
uart[8250],0x<addr>[,options]
        Start an early, polled-mode console on the 8250/16550
        UART at the specified I/O port or MMIO address.
        MMIO inter-register address stride is either 8-bit
        (mmio) or 32-bit (mmio32 or mmio32be).
        If none of [io|mmio|mmio32|mmio32be], <addr> is assumed
        to be equivalent to `mmio'. `options' are specified
        in the same format described for ``console=ttyS<n>''; if
        unspecified, the h/w is not initialized.

pl011,<addr>
pl011,mmio32,<addr>
        Start an early, polled-mode console on a pl011 serial
        port at the specified address. The pl011 serial port
        must already be setup and configured. Options are not
        yet supported.  If `mmio32' is specified, then only
        the driver will use only 32-bit accessors to read/write
        the device registers.

meson,<addr>
        Start an early, polled-mode console on a meson serial
        port at the specified address. The serial port must
        already be setup and configured. Options are not yet
        supported.

msm_serial,<addr>
        Start an early, polled-mode console on an msm serial
        port at the specified address. The serial port
        must already be setup and configured. Options are not
        yet supported.

msm_serial_dm,<addr>
        Start an early, polled-mode console on an msm serial
        dm port at the specified address. The serial port
        must already be setup and configured. Options are not
        yet supported.

owl,<addr>
        Start an early, polled-mode console on a serial port
        of an Actions Semi SoC, such as S500 or S900, at the
        specified address. The serial port must already be
        setup and configured. Options are not yet supported.

smh     Use ARM semihosting calls for early console.

s3c2410,<addr>
s3c2412,<addr>
s3c2440,<addr>
s3c6400,<addr>
s5pv210,<addr>
exynos4210,<addr>
        Use early console provided by serial driver available
```

```
                on Samsung SoCs, requires selecting proper type and
                a correct base address of the selected UART port. The
                serial port must already be setup and configured.
                Options are not yet supported.

        lantiq,<addr>
                Start an early, polled-mode console on a lantiq serial
                (lqasc) port at the specified address. The serial port
                must already be setup and configured. Options are not
                yet supported.

        lpuart,<addr>
        lpuart32,<addr>
                Use early console provided by Freescale LP UART driver
                found on Freescale Vybrid and QorIQ LS1021A processors.
                A valid base address must be provided, and the serial
                port must already be setup and configured.

        ar3700_uart,<addr>
                Start an early, polled-mode console on the
                Armada 3700 serial port at the specified
                address. The serial port must already be setup
                and configured. Options are not yet supported.

earlyprintk=    [X86,SH,BLACKFIN,ARM,M68k,S390]
                earlyprintk=vga
                earlyprintk=efi
                earlyprintk=sclp
                earlyprintk=xen
                earlyprintk=serial[,ttySn[,baudrate]]
                earlyprintk=serial[,0x...[,baudrate]]
                earlyprintk=ttySn[,baudrate]
                earlyprintk=dbgp[debugController#]
                earlyprintk=pciserial,bus:device.function[,baudrate]
                earlyprintk=xdbc[xhciController#]

                earlyprintk is useful when the kernel crashes before
                the normal console is initialized. It is not enabled by
                default because it has some cosmetic problems.

                Append '',keep'' to not disable it when the real console
                takes over.

                Only one of vga, efi, serial, or usb debug port can
                be used at a time.

                Currently only ttyS0 and ttyS1 may be specified by
                name.  Other I/O ports may be explicitly specified
                on some architectures (x86 and arm at least) by
                replacing ttySn with an I/O port address, like this:
                        earlyprintk=serial,0x1008,115200
                You can find the port for a given device in
                /proc/tty/driver/serial:
                        2: uart:ST16650V2 port:00001008 irq:18 ...

                Interaction with the standard serial driver is not
                very good.
```

```
                The VGA and EFI output is eventually overwritten by
                the real console.

                The xen output can only be used by Xen PV guests.

                The sclp output can only be used on s390.

edac_report=    [HW,EDAC] Control how to report EDAC event
                Format: {``on'' | ``off'' | ``force''}
                on: enable EDAC to report H/W event. May be overridden
                by other higher priority error reporting module.
                off: disable H/W event reporting through EDAC.
                force: enforce the use of EDAC to report H/W event.
                default: on.

ekgdboc=        [X86,KGDB] Allow early kernel console debugging
                ekgdboc=kbd

                This is designed to be used in conjunction with
                the boot argument: earlyprintk=vga

edd=            [EDD]
                Format: {``off'' | ``on'' | ``skip[mbr]''}

efi=            [EFI]
                Format: { ``old_map'', ``nochunk'', ``noruntime'', ``debug'' }
                old_map [X86-64]: switch to the old ioremap-based EFI
                runtime services mapping. 32-bit still uses this one by
                default.
                nochunk: disable reading files in ``chunks'' in the EFI
                boot stub, as chunking can cause problems with some
                firmware implementations.
                noruntime : disable EFI runtime services support
                debug: enable misc debug output

efi_no_storage_paranoia [EFI; X86]
                Using this parameter you can use more than 50% of
                your efi variable storage. Use this parameter only if
                you are really sure that your UEFI does sane gc and
                fulfills the spec otherwise your board may brick.

efi_fake_mem=   nn[KMG]@ss[KMG]:aa[,nn[KMG]@ss[KMG]:aa,..] [EFI; X86]
                Add arbitrary attribute to specific memory range by
                updating original EFI memory map.
                Region of memory which aa attribute is added to is
                from ss to ss+nn.
                If efi_fake_mem=2G@4G:0x10000,2G@0x10a0000000:0x10000
                is specified, EFI_MEMORY_MORE_RELIABLE(0x10000)
                attribute is added to range 0x100000000-0x180000000 and
                0x10a0000000-0x1120000000.

                Using this parameter you can do debugging of EFI memmap
                related feature. For example, you can do debugging of
                Address Range Mirroring feature even if your box
                doesn't support it.

efivar_ssdt=    [EFI; X86] Name of an EFI variable that contains an SSDT
                that is to be dynamically loaded by Linux. If there are
```

```
                multiple variables with the same name but with different
                vendor GUIDs, all of them will be loaded. See
                Documentation/acpi/ssdt-overlays.txt for details.


eisa_irq_edge=  [PARISC,HW]
                See header of drivers/parisc/eisa.c.

elanfreq=       [X86-32]
                See comment before function elanfreq_setup() in
                arch/x86/kernel/cpu/cpufreq/elanfreq.c.

elevator=       [IOSCHED]
                Format: {``cfq'' | ``deadline'' | ``noop''}
                See Documentation/block/cfq-iosched.txt and
                Documentation/block/deadline-iosched.txt for details.

elfcorehdr=[size[KMG]@]offset[KMG] [IA64,PPC,SH,X86,S390]
                Specifies physical address of start of kernel core
                image elf header and optionally the size. Generally
                kexec loader will pass this option to capture kernel.
                See Documentation/kdump/kdump.txt for details.

enable_mtrr_cleanup [X86]
                The kernel tries to adjust MTRR layout from continuous
                to discrete, to make X server driver able to add WB
                entry later. This parameter enables that.

enable_timer_pin_1 [X86]
                Enable PIN 1 of APIC timer
                Can be useful to work around chipset bugs
                (in particular on some ATI chipsets).
                The kernel tries to set a reasonable default.

enforcing       [SELINUX] Set initial enforcing status.
                Format: {``0'' | ``1''}
                See security/selinux/Kconfig help text.
                0 -- permissive (log only, no denials).
                1 -- enforcing (deny and log).
                Default value is 0.
                Value can be changed at runtime via /selinux/enforce.

erst_disable    [ACPI]
                Disable Error Record Serialization Table (ERST)
                support.

ether=          [HW,NET] Ethernet cards parameters
                This option is obsoleted by the ``netdev='' option, which
                has equivalent usage. See its documentation for details.

evm=            [EVM]
                Format: { ``fix'' }
                Permit `security.evm' to be updated regardless of
                current integrity status.

failslab=
fail_page_alloc=
fail_make_request=[KNL]
```

```
                General fault injection mechanism.
                Format: <interval>,<probability>,<space>,<times>
                See also Documentation/fault-injection/.

floppy=         [HW]
                See Documentation/blockdev/floppy.txt.

force_pal_cache_flush
                [IA-64] Avoid check_sal_cache_flush which may hang on
                buggy SAL_CACHE_FLUSH implementations. Using this
                parameter will force ia64_sal_cache_flush to call
                ia64_pal_cache_flush instead of SAL_CACHE_FLUSH.

forcepae [X86-32]
                Forcefully enable Physical Address Extension (PAE).
                Many Pentium M systems disable PAE but may have a
                functionally usable PAE implementation.
                Warning: use of this parameter will taint the kernel
                and may cause unknown problems.

ftrace=[tracer]
                [FTRACE] will set and start the specified tracer
                as early as possible in order to facilitate early
                boot debugging.

ftrace_dump_on_oops[=orig_cpu]
                [FTRACE] will dump the trace buffers on oops.
                If no parameter is passed, ftrace will dump
                buffers of all CPUs, but if you pass orig_cpu, it will
                dump only the buffer of the CPU that triggered the
                oops.

ftrace_filter=[function-list]
                [FTRACE] Limit the functions traced by the function
                tracer at boot up. function-list is a comma separated
                list of functions. This list can be changed at run
                time by the set_ftrace_filter file in the debugfs
                tracing directory.

ftrace_notrace=[function-list]
                [FTRACE] Do not trace the functions specified in
                function-list. This list can be changed at run time
                by the set_ftrace_notrace file in the debugfs
                tracing directory.

ftrace_graph_filter=[function-list]
                [FTRACE] Limit the top level callers functions traced
                by the function graph tracer at boot up.
                function-list is a comma separated list of functions
                that can be changed at run time by the
                set_graph_function file in the debugfs tracing directory.

ftrace_graph_notrace=[function-list]
                [FTRACE] Do not trace from the functions specified in
                function-list.  This list is a comma separated list of
                functions that can be changed at run time by the
                set_graph_notrace file in the debugfs tracing directory.
```

```
ftrace_graph_max_depth=<uint>
                [FTRACE] Used with the function graph tracer. This is
                the max depth it will trace into a function. This value
                can be changed at run time by the max_graph_depth file
                in the tracefs tracing directory. default: 0 (no limit)

gamecon.map[2|3]=
                [HW,JOY] Multisystem joystick and NES/SNES/PSX pad
                support via parallel port (up to 5 devices per port)
                Format: <port#>,<pad1>,<pad2>,<pad3>,<pad4>,<pad5>
                See also Documentation/input/joystick-parport.txt

gamma=          [HW,DRM]

gart_fix_e820=  [X86_64] disable the fix e820 for K8 GART
                Format: off | on
                default: on

gcov_persist=   [GCOV] When non-zero (default), profiling data for
                kernel modules is saved and remains accessible via
                debugfs, even when the module is unloaded/reloaded.
                When zero, profiling data is discarded and associated
                debugfs files are removed at module unload time.

goldfish        [X86] Enable the goldfish android emulator platform.
                Don't use this when you are not running on the
                android emulator

gpt             [EFI] Forces disk with valid GPT signature but
                invalid Protective MBR to be treated as GPT. If the
                primary GPT is corrupted, it enables the backup/alternate
                GPT to be used instead.

grcan.enable0=  [HW] Configuration of physical interface 0. Determines
                the ``Enable 0'' bit of the configuration register.
                Format: 0 | 1
                Default: 0
grcan.enable1=  [HW] Configuration of physical interface 1. Determines
                the ``Enable 0'' bit of the configuration register.
                Format: 0 | 1
                Default: 0
grcan.select=   [HW] Select which physical interface to use.
                Format: 0 | 1
                Default: 0
grcan.txsize=   [HW] Sets the size of the tx buffer.
                Format: <unsigned int> such that (txsize & ~0x1fffc0) == 0.
                Default: 1024
grcan.rxsize=   [HW] Sets the size of the rx buffer.
                Format: <unsigned int> such that (rxsize & ~0x1fffc0) == 0.
                Default: 1024

gpio-mockup.gpio_mockup_ranges
                [HW] Sets the ranges of gpiochip of for this device.
                Format: <start1>,<end1>,<start2>,<end2>...

hardlockup_all_cpu_backtrace=
                [KNL] Should the hard-lockup detector generate
                backtraces on all cpus.
```

```
                Format: <integer>

hashdist=       [KNL,NUMA] Large hashes allocated during boot
                are distributed across NUMA nodes.  Defaults on
                for 64-bit NUMA, off otherwise.
                Format: 0 | 1 (for off | on)

hcl=            [IA-64] SGI's Hardware Graph compatibility layer

hd=             [EIDE] (E)IDE hard drive subsystem geometry
                Format: <cyl>,<head>,<sect>

hest_disable    [ACPI]
                Disable Hardware Error Source Table (HEST) support;
                corresponding firmware-first mode error processing
                logic will be disabled.

highmem=nn[KMG] [KNL,BOOT] forces the highmem zone to have an exact
                size of <nn>. This works even on boxes that have no
                highmem otherwise. This also works to reduce highmem
                size on bigger boxes.

highres=        [KNL] Enable/disable high resolution timer mode.
                Valid parameters: ``on'', ``off''
                Default: ``on''

hisax=          [HW,ISDN]
                See Documentation/isdn/README.HiSax.

hlt             [BUGS=ARM,SH]

hpet=           [X86-32,HPET] option to control HPET usage
                Format: { enable (default) | disable | force |
                        verbose }
                disable: disable HPET and use PIT instead
                force: allow force enabled of undocumented chips (ICH4,
                        VIA, nVidia)
                verbose: show contents of HPET registers during setup

hpet_mmap=      [X86, HPET_MMAP] Allow userspace to mmap HPET
                registers.  Default set by CONFIG_HPET_MMAP_DEFAULT.

hugepages=      [HW,X86-32,IA-64] HugeTLB pages to allocate at boot.
hugepagesz=     [HW,IA-64,PPC,X86-64] The size of the HugeTLB pages.
                On x86-64 and powerpc, this option can be specified
                multiple times interleaved with hugepages= to reserve
                huge pages of different sizes. Valid pages sizes on
                x86-64 are 2M (when the CPU supports ``pse'') and 1G
                (when the CPU supports the ``pdpe1gb'' cpuinfo flag).

hvc_iucv=       [S390] Number of z/VM IUCV hypervisor console (HVC)
                       terminal devices. Valid values: 0..8
hvc_iucv_allow= [S390] Comma-separated list of z/VM user IDs.
                       If specified, z/VM IUCV HVC accepts connections
                       from listed z/VM user IDs only.

hwthread_map=   [METAG] Comma-separated list of Linux cpu id to
                        hardware thread id mappings.
```

```
                        Format: <cpu>:<hwthread>

keep_bootcon    [KNL]
                Do not unregister boot console at start. This is only
                useful for debugging when something happens in the window
                between unregistering the boot console and initializing
                the real console.

i2c_bus=        [HW] Override the default board specific I2C bus speed
                     or register an additional I2C bus that is not
                     registered from board initialization code.
                     Format:
                     <bus_id>,<clkrate>

i8042.debug     [HW] Toggle i8042 debug mode
i8042.unmask_kbd_data
                [HW] Enable printing of interrupt data from the KBD port
                     (disabled by default, and as a pre-condition
                     requires that i8042.debug=1 be enabled)
i8042.direct    [HW] Put keyboard port into non-translated mode
i8042.dumbkbd   [HW] Pretend that controller can only read data from
                     keyboard and cannot control its state
                     (Don't attempt to blink the leds)
i8042.noaux     [HW] Don't check for auxiliary (== mouse) port
i8042.nokbd     [HW] Don't check/create keyboard port
i8042.noloop    [HW] Disable the AUX Loopback command while probing
                     for the AUX port
i8042.nomux     [HW] Don't check presence of an active multiplexing
                     controller
i8042.nopnp     [HW] Don't use ACPIPnP / PnPBIOS to discover KBD/AUX
                     controllers
i8042.notimeout [HW] Ignore timeout condition signalled by controller
i8042.reset     [HW] Reset the controller during init, cleanup and
                     suspend-to-ram transitions, only during s2r
                     transitions, or never reset
                Format: { 1 | Y | y | 0 | N | n }
                1, Y, y: always reset controller
                0, N, n: don't ever reset controller
                Default: only on s2r transitions on x86; most other
                architectures force reset to be always executed
i8042.unlock    [HW] Unlock (ignore) the keylock
i8042.kbdreset  [HW] Reset device connected to KBD port

i810=           [HW,DRM]

i8k.ignore_dmi  [HW] Continue probing hardware even if DMI data
                indicates that the driver is running on unsupported
                hardware.
i8k.force       [HW] Activate i8k driver even if SMM BIOS signature
                does not match list of supported models.
i8k.power_status
                [HW] Report power status in /proc/i8k
                (disabled by default)
i8k.restricted  [HW] Allow controlling fans only if SYS_ADMIN
                capability is set.

i915.invert_brightness=
                [DRM] Invert the sense of the variable that is used to
```

```
                set the brightness of the panel backlight. Normally a
                brightness value of 0 indicates backlight switched off,
                and the maximum of the brightness value sets the backlight
                to maximum brightness. If this parameter is set to 0
                (default) and the machine requires it, or this parameter
                is set to 1, a brightness value of 0 sets the backlight
                to maximum brightness, and the maximum of the brightness
                value switches the backlight off.
                -1 -- never invert brightness
                 0 -- machine default
                 1 -- force brightness inversion

icn=            [HW,ISDN]
                Format: <io>[,<membase>[,<icn_id>[,<icn_id2>]]]

ide-core.nodma= [HW] (E)IDE subsystem
                Format: =0.0 to prevent dma on hda, =0.1 hdb =1.0 hdc
                .vlb_clock .pci_clock .noflush .nohpa .noprobe .nowerr
                .cdrom .chs .ignore_cable are additional options
                See Documentation/ide/ide.txt.

ide-generic.probe-mask= [HW] (E)IDE subsystem
                Format: <int>
                Probe mask for legacy ISA IDE ports.  Depending on
                platform up to 6 ports are supported, enabled by
                setting corresponding bits in the mask to 1.  The
                default value is 0x0, which has a special meaning.
                On systems that have PCI, it triggers scanning the
                PCI bus for the first and the second port, which
                are then probed.  On systems without PCI the value
                of 0x0 enables probing the two first ports as if it
                was 0x3.

ide-pci-generic.all-generic-ide [HW] (E)IDE subsystem
                Claim all unknown PCI IDE storage controllers.

idle=           [X86]
                Format: idle=poll, idle=halt, idle=nomwait
                Poll forces a polling idle loop that can slightly
                improve the performance of waking up a idle CPU, but
                will use a lot of power and make the system run hot.
                Not recommended.
                idle=halt: Halt is forced to be used for CPU idle.
                In such case C2/C3 won't be used again.
                idle=nomwait: Disable mwait for CPU C-states

ieee754=        [MIPS] Select IEEE Std 754 conformance mode
                Format: { strict | legacy | 2008 | relaxed }
                Default: strict

                Choose which programs will be accepted for execution
                based on the IEEE 754 NaN encoding(s) supported by
                the FPU and the NaN encoding requested with the value
                of an ELF file header flag individually set by each
                binary.  Hardware implementations are permitted to
                support either or both of the legacy and the 2008 NaN
                encoding mode.
```

```
                Available settings are as follows:
                strict  accept binaries that request a NaN encoding
                        supported by the FPU
                legacy  only accept legacy-NaN binaries, if supported
                        by the FPU
                2008    only accept 2008-NaN binaries, if supported
                        by the FPU
                relaxed accept any binaries regardless of whether
                        supported by the FPU

                The FPU emulator is always able to support both NaN
                encodings, so if no FPU hardware is present or it has
                been disabled with `nofpu', then the settings of
                `legacy' and `2008' strap the emulator accordingly,
                `relaxed' straps the emulator for both legacy-NaN and
                2008-NaN, whereas `strict' enables legacy-NaN only on
                legacy processors and both NaN encodings on MIPS32 or
                MIPS64 CPUs.

                The setting for ABS.fmt/NEG.fmt instruction execution
                mode generally follows that for the NaN encoding,
                except where unsupported by hardware.

ignore_loglevel [KNL]
                Ignore loglevel setting - this will print /all/
                kernel messages to the console. Useful for debugging.
                We also add it as printk module parameter, so users
                could change it dynamically, usually by
                /sys/module/printk/parameters/ignore_loglevel.

ignore_rlimit_data
                Ignore RLIMIT_DATA setting for data mappings,
                print warning at first misuse.  Can be changed via
                /sys/module/kernel/parameters/ignore_rlimit_data.

ihash_entries=  [KNL]
                Set number of hash buckets for inode cache.

ima_appraise=   [IMA] appraise integrity measurements
                Format: { ``off'' | ``enforce'' | ``fix'' | ``log'' }
                default: ``enforce''

ima_appraise_tcb [IMA]
                The builtin appraise policy appraises all files
                owned by uid=0.

ima_canonical_fmt [IMA]
                Use the canonical format for the binary runtime
                measurements, instead of host native format.

ima_hash=       [IMA]
                Format: { md5 | sha1 | rmd160 | sha256 | sha384
                          | sha512 | ... }
                default: ``sha1''

                The list of supported hash algorithms is defined
                in crypto/hash_info.h.
```

```
ima_policy=     [IMA]
                The builtin policies to load during IMA setup.
                Format: ``tcb | appraise_tcb | secure_boot''

                The ``tcb'' policy measures all programs exec'd, files
                mmap'd for exec, and all files opened with the read
                mode bit set by either the effective uid (euid=0) or
                uid=0.

                The ``appraise_tcb'' policy appraises the integrity of
                all files owned by root. (This is the equivalent
                of ima_appraise_tcb.)

                The ``secure_boot'' policy appraises the integrity
                of files (eg. kexec kernel image, kernel modules,
                firmware, policy, etc) based on file signatures.

ima_tcb         [IMA] Deprecated.  Use ima_policy= instead.
                Load a policy which meets the needs of the Trusted
                Computing Base.  This means IMA will measure all
                programs exec'd, files mmap'd for exec, and all files
                opened for read by uid=0.

ima_template=   [IMA]
                Select one of defined IMA measurements template formats.
                Formats: { ``ima'' | ``ima-ng'' | ``ima-sig'' }
                Default: ``ima-ng''

ima_template_fmt=
                [IMA] Define a custom template format.
                Format: { ``field1|...|fieldN'' }

ima.ahash_minsize= [IMA] Minimum file size for asynchronous hash usage
                Format: <min_file_size>
                Set the minimal file size for using asynchronous hash.
                If left unspecified, ahash usage is disabled.

                ahash performance varies for different data sizes on
                different crypto accelerators. This option can be used
                to achieve the best performance for a particular HW.

ima.ahash_bufsize= [IMA] Asynchronous hash buffer size
                Format: <bufsize>
                Set hashing buffer size. Default: 4k.

                ahash performance varies for different chunk sizes on
                different crypto accelerators. This option can be used
                to achieve best performance for particular HW.

init=           [KNL]
                Format: <full_path>
                Run specified binary instead of /sbin/init as init
                process.

initcall_debug  [KNL] Trace initcalls as they are executed.  Useful
                for working out where the kernel is dying during
                startup.
```

```
initcall_blacklist=  [KNL] Do not execute a comma-separated list of
                initcall functions.  Useful for debugging built-in
                modules and initcalls.

initrd=         [BOOT] Specify the location of the initial ramdisk

init_pkru=      [x86] Specify the default memory protection keys rights
                register contents for all processes.  0x55555554 by
                default (disallow access to all but pkey 0).  Can
                override in debugfs after boot.

inport.irq=     [HW] Inport (ATI XL and Microsoft) busmouse driver
                Format: <irq>

int_pln_enable  [x86] Enable power limit notification interrupt

integrity_audit=[IMA]
                Format: { ``0'' | ``1'' }
                0 -- basic integrity auditing messages. (Default)
                1 -- additional integrity auditing messages.

intel_iommu=    [DMAR] Intel IOMMU driver (DMAR) option
        on
                Enable intel iommu driver.
        off
                Disable intel iommu driver.
        igfx_off [Default Off]
                By default, gfx is mapped as normal device. If a gfx
                device has a dedicated DMAR unit, the DMAR unit is
                bypassed by not enabling DMAR with this option. In
                this case, gfx device will use physical address for
                DMA.
        forcedac [x86_64]
                With this option iommu will not optimize to look
                for io virtual address below 32-bit forcing dual
                address cycle on pci bus for cards supporting greater
                than 32-bit addressing. The default is to look
                for translation below 32-bit and if not available
                then look in the higher range.
        strict [Default Off]
                With this option on every unmap_single operation will
                result in a hardware IOTLB flush operation as opposed
                to batching them for performance.
        sp_off [Default Off]
                By default, super page will be supported if Intel IOMMU
                has the capability. With this option, super page will
                not be supported.
        ecs_off [Default Off]
                By default, extended context tables will be supported if
                the hardware advertises that it has support both for the
                extended tables themselves, and also PASID support. With
                this option set, extended tables will not be used even
                on hardware which claims to support them.
        tboot_noforce [Default Off]
                Do not force the Intel IOMMU enabled under tboot.
                By default, tboot will force Intel IOMMU on, which
                could harm performance of some high-throughput
                devices like 40GBit network cards, even if identity
```

```
                mapping is enabled.
                Note that using this option lowers the security
                provided by tboot because it makes the system
                vulnerable to DMA attacks.

intel_idle.max_cstate=  [KNL,HW,ACPI,X86]
                0       disables intel_idle and fall back on acpi_idle.
                1 to 9  specify maximum depth of C-state.

intel_pstate=  [X86]
               disable
                 Do not enable intel_pstate as the default
                 scaling driver for the supported processors
               passive
                 Use intel_pstate as a scaling driver, but configure it
                 to work with generic cpufreq governors (instead of
                 enabling its internal governor).  This mode cannot be
                 used along with the hardware-managed P-states (HWP)
                 feature.
               force
                 Enable intel_pstate on systems that prohibit it by default
                 in favor of acpi-cpufreq. Forcing the intel_pstate driver
                 instead of acpi-cpufreq may disable platform features, such
                 as thermal controls and power capping, that rely on ACPI
                 P-States information being indicated to OSPM and therefore
                 should be used with caution. This option does not work with
                 processors that aren't supported by the intel_pstate driver
                 or on platforms that use pcc-cpufreq instead of acpi-cpufreq.
               no_hwp
                 Do not enable hardware P state control (HWP)
                 if available.
        hwp_only
                Only load intel_pstate on systems which support
                hardware P state control (HWP) if available.
        support_acpi_ppc
                Enforce ACPI _PPC performance limits. If the Fixed ACPI
                Description Table, specifies preferred power management
                profile as ``Enterprise Server'' or ``Performance Server'',
                then this feature is turned on by default.
        per_cpu_perf_limits
                Allow per-logical-CPU P-State performance control limits using
                cpufreq sysfs interface

intremap=       [X86-64, Intel-IOMMU]
                on      enable Interrupt Remapping (default)
                off     disable Interrupt Remapping
                nosid   disable Source ID checking
                no_x2apic_optout
                        BIOS x2APIC opt-out request will be ignored
                nopost  disable Interrupt Posting

iomem=          Disable strict checking of access to MMIO memory
        strict  regions from userspace.
        relaxed

iommu=          [x86]
        off
        force
```

```
        noforce
        biomerge
        panic
        nopanic
        merge
        nomerge
        forcesac
        soft
        pt              [x86, IA-64]
        nobypass        [PPC/POWERNV]
                Disable IOMMU bypass, using IOMMU for PCI devices.

iommu.passthrough=
                [ARM64] Configure DMA to bypass the IOMMU by default.
                Format: { ``0'' | ``1'' }
                0 - Use IOMMU translation for DMA.
                1 - Bypass the IOMMU for DMA.
                unset - Use IOMMU translation for DMA.

io7=            [HW] IO7 for Marvel based alpha systems
                See comment before marvel_specify_io7 in
                arch/alpha/kernel/core_marvel.c.

io_delay=       [X86] I/O delay method
        0x80
                Standard port 0x80 based delay
        0xed
                Alternate port 0xed based delay (needed on some systems)
        udelay
                Simple two microseconds delay
        none
                No delay

ip=             [IP_PNP]
                See Documentation/filesystems/nfs/nfsroot.txt.

irqaffinity=    [SMP] Set the default irq affinity mask
                The argument is a cpu list, as described above.

irqfixup        [HW]
                When an interrupt is not handled search all handlers
                for it. Intended to get systems with badly broken
                firmware running.

irqpoll         [HW]
                When an interrupt is not handled search all handlers
                for it. Also check all handlers each timer
                interrupt. Intended to get systems with badly broken
                firmware running.

isapnp=         [ISAPNP]
                Format: <RDP>,<reset>,<pci_scan>,<verbosity>

isolcpus=       [KNL,SMP] Isolate CPUs from the general scheduler.
                The argument is a cpu list, as described above.

                This option can be used to specify one or more CPUs
                to isolate from the general SMP balancing and scheduling
```

```
                algorithms. You can move a process onto or off an
                ``isolated'' CPU via the CPU affinity syscalls or cpuset.
                <cpu number> begins at 0 and the maximum value is
                ``number of CPUs in system - 1''.

                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.

iucv=           [HW,NET]

ivrs_ioapic     [HW,X86_64]
                Provide an override to the IOAPIC-ID<->DEVICE-ID
                mapping provided in the IVRS ACPI table. For
                example, to map IOAPIC-ID decimal 10 to
                PCI device 00:14.0 write the parameter as:
                        ivrs_ioapic[10]=00:14.0

ivrs_hpet       [HW,X86_64]
                Provide an override to the HPET-ID<->DEVICE-ID
                mapping provided in the IVRS ACPI table. For
                example, to map HPET-ID decimal 0 to
                PCI device 00:14.0 write the parameter as:
                        ivrs_hpet[0]=00:14.0

ivrs_acpihid    [HW,X86_64]
                Provide an override to the ACPI-HID:UID<->DEVICE-ID
                mapping provided in the IVRS ACPI table. For
                example, to map UART-HID:UID AMD0020:0 to
                PCI device 00:14.5 write the parameter as:
                        ivrs_acpihid[00:14.5]=AMD0020:0

js=             [HW,JOY] Analog joystick
                See Documentation/input/joystick.txt.

nokaslr         [KNL]
                When CONFIG_RANDOMIZE_BASE is set, this disables
                kernel and module base offset ASLR (Address Space
                Layout Randomization).

kasan_multi_shot
                [KNL] Enforce KASAN (Kernel Address Sanitizer) to print
                report on every invalid memory access. Without this
                parameter KASAN will print report only for the first
                invalid access.

keepinitrd      [HW,ARM]

kernelcore=     [KNL,X86,IA-64,PPC]
                Format: nn[KMGTPE] | ``mirror''
                This parameter
                specifies the amount of memory usable by the kernel
                for non-movable allocations.  The requested amount is
                spread evenly throughout all nodes in the system. The
                remaining memory in each node is used for Movable
                pages. In the event, a node is too small to have both
                kernelcore and Movable pages, kernelcore pages will
```

```
                take priority and other nodes will have a larger number
                of Movable pages.  The Movable zone is used for the
                allocation of pages that may be reclaimed or moved
                by the page migration subsystem.  This means that
                HugeTLB pages may not be allocated from this zone.
                Note that allocations like PTEs-from-HighMem still
                use the HighMem zone if it exists, and the Normal
                zone if it does not.

                Instead of specifying the amount of memory (nn[KMGTPE]),
                you can specify ``mirror'' option. In case ``mirror''
                option is specified, mirrored (reliable) memory is used
                for non-movable allocations and remaining memory is used
                for Movable pages. nn[KMGTPE] and ``mirror'' are exclusive,
                so you can NOT specify nn[KMGTPE] and ``mirror'' at the same
                time.

    kgdbdbgp=   [KGDB,HW] kgdb over EHCI usb debug port.
                Format: <Controller#>[,poll interval]
                The controller # is the number of the ehci usb debug
                port as it is probed via PCI.  The poll interval is
                optional and is the number seconds in between
                each poll cycle to the debug port in case you need
                the functionality for interrupting the kernel with
                gdb or control-c on the dbgp connection.  When
                not using this parameter you use sysrq-g to break into
                the kernel debugger.

    kgdboc=     [KGDB,HW] kgdb over consoles.
                Requires a tty driver that supports console polling,
                or a supported polling keyboard driver (non-usb).
                 Serial only format: <serial_device>[,baud]
                 keyboard only format: kbd
                 keyboard and serial format: kbd,<serial_device>[,baud]
                Optional Kernel mode setting:
                 kms, kbd format: kms,kbd
                 kms, kbd and serial format: kms,kbd,<ser_dev>[,baud]

    kgdbwait    [KGDB] Stop kernel execution and enter the
                kernel debugger at the earliest opportunity.

    kmac=       [MIPS] korina ethernet MAC address.
                Configure the RouterBoard 532 series on-chip
                Ethernet adapter MAC address.

    kmemleak=   [KNL] Boot-time kmemleak enable/disable
                Valid arguments: on, off
                Default: on
                Built with CONFIG_DEBUG_KMEMLEAK_DEFAULT_OFF=y,
                the default is off.

    kmemcheck=  [X86] Boot-time kmemcheck enable/disable/one-shot mode
                Valid arguments: 0, 1, 2
                kmemcheck=0 (disabled)
                kmemcheck=1 (enabled)
                kmemcheck=2 (one-shot mode)
                Default: 2 (one-shot mode)
```

```
kvm.ignore_msrs=[KVM] Ignore guest accesses to unhandled MSRs.
                Default is 0 (don't ignore, but inject #GP)

kvm.mmu_audit=  [KVM] This is a R/W parameter which allows audit
                KVM MMU at runtime.
                Default is 0 (off)

kvm-amd.nested= [KVM,AMD] Allow nested virtualization in KVM/SVM.
                Default is 1 (enabled)

kvm-amd.npt=    [KVM,AMD] Disable nested paging (virtualized MMU)
                for all guests.
                Default is 1 (enabled) if in 64-bit or 32-bit PAE mode.

kvm-arm.vgic_v3_group0_trap=
                [KVM,ARM] Trap guest accesses to GICv3 group-0
                system registers

kvm-arm.vgic_v3_group1_trap=
                [KVM,ARM] Trap guest accesses to GICv3 group-1
                system registers

kvm-arm.vgic_v3_common_trap=
                [KVM,ARM] Trap guest accesses to GICv3 common
                system registers

kvm-intel.ept=  [KVM,Intel] Disable extended page tables
                (virtualized MMU) support on capable Intel chips.
                Default is 1 (enabled)

kvm-intel.emulate_invalid_guest_state=
                [KVM,Intel] Enable emulation of invalid guest states
                Default is 0 (disabled)

kvm-intel.flexpriority=
                [KVM,Intel] Disable FlexPriority feature (TPR shadow).
                Default is 1 (enabled)

kvm-intel.nested=
                [KVM,Intel] Enable VMX nesting (nVMX).
                Default is 0 (disabled)

kvm-intel.unrestricted_guest=
                [KVM,Intel] Disable unrestricted guest feature
                (virtualized real and unpaged mode) on capable
                Intel chips. Default is 1 (enabled)

kvm-intel.vpid= [KVM,Intel] Disable Virtual Processor Identification
                feature (tagged TLBs) on capable Intel chips.
                Default is 1 (enabled)

l2cr=           [PPC]

l3cr=           [PPC]

lapic           [X86-32,APIC] Enable the local APIC even if BIOS
                disabled it.
```

```
lapic=          [x86,APIC] ``notscdeadline'' Do not use TSC deadline
                value for LAPIC timer one-shot implementation. Default
                back to the programmable timer unit in the LAPIC.

lapic_timer_c2_ok       [X86,APIC] trust the local apic timer
                in C2 power state.

libata.dma=     [LIBATA] DMA control
                libata.dma=0      Disable all PATA and SATA DMA
                libata.dma=1      PATA and SATA Disk DMA only
                libata.dma=2      ATAPI (CDROM) DMA only
                libata.dma=4      Compact Flash DMA only
                Combinations also work, so libata.dma=3 enables DMA
                for disks and CDROMs, but not CFs.

libata.ignore_hpa=      [LIBATA] Ignore HPA limit
                libata.ignore_hpa=0       keep BIOS limits (default)
                libata.ignore_hpa=1       ignore limits, using full disk

libata.noacpi   [LIBATA] Disables use of ACPI in libata suspend/resume
                when set.
                Format: <int>

libata.force=   [LIBATA] Force configurations.  The format is comma
                separated list of ``[ID:]VAL'' where ID is
                PORT[.DEVICE].  PORT and DEVICE are decimal numbers
                matching port, link or device.  Basically, it matches
                the ATA ID string printed on console by libata.  If
                the whole ID part is omitted, the last PORT and DEVICE
                values are used.  If ID hasn't been specified yet, the
                configuration applies to all ports, links and devices.

                If only DEVICE is omitted, the parameter applies to
                the port and all links and devices behind it.  DEVICE
                number of 0 either selects the first device or the
                first fan-out link behind PMP device.  It does not
                select the host link.  DEVICE number of 15 selects the
                host link and device attached to it.

                The VAL specifies the configuration to force.  As long
                as there's no ambiguity shortcut notation is allowed.
                For example, both 1.5 and 1.5G would work for 1.5Gbps.
                The following configurations can be forced.

                * Cable type: 40c, 80c, short40c, unk, ign or sata.
                  Any ID with matching PORT is used.

                * SATA link speed limit: 1.5Gbps or 3.0Gbps.

                * Transfer mode: pio[0-7], mwdma[0-4] and udma[0-7].
                  udma[/][16,25,33,44,66,100,133] notation is also
                  allowed.

                * [no]ncq: Turn on or off NCQ.

                * [no]ncqtrim: Turn off queued DSM TRIM.

                * nohrst, nosrst, norst: suppress hard, soft
```

```
                  and both resets.

                * rstonce: only attempt one reset during
                  hot-unplug link recovery

                * dump_id: dump IDENTIFY data.

                * atapi_dmadir: Enable ATAPI DMADIR bridge support

                * disable: Disable this device.

                If there are multiple matching configurations changing
                the same attribute, the last one is used.

memblock=debug  [KNL] Enable memblock debug messages.

load_ramdisk=   [RAM] List of ramdisks to load from floppy
                See Documentation/blockdev/ramdisk.txt.

lockd.nlm_grace_period=P  [NFS] Assign grace period.
                Format: <integer>

lockd.nlm_tcpport=N     [NFS] Assign TCP port.
                Format: <integer>

lockd.nlm_timeout=T     [NFS] Assign timeout value.
                Format: <integer>

lockd.nlm_udpport=M     [NFS] Assign UDP port.
                Format: <integer>

locktorture.nreaders_stress= [KNL]
                Set the number of locking read-acquisition kthreads.
                Defaults to being automatically set based on the
                number of online CPUs.

locktorture.nwriters_stress= [KNL]
                Set the number of locking write-acquisition kthreads.

locktorture.onoff_holdoff= [KNL]
                Set time (s) after boot for CPU-hotplug testing.

locktorture.onoff_interval= [KNL]
                Set time (s) between CPU-hotplug operations, or
                zero to disable CPU-hotplug testing.

locktorture.shuffle_interval= [KNL]
                Set task-shuffle interval (jiffies).  Shuffling
                tasks allows some CPUs to go into dyntick-idle
                mode during the locktorture test.

locktorture.shutdown_secs= [KNL]
                Set time (s) after boot system shutdown.  This
                is useful for hands-off automated testing.

locktorture.stat_interval= [KNL]
                Time (s) between statistics printk()s.
```

```
locktorture.stutter= [KNL]
                Time (s) to stutter testing, for example,
                specifying five seconds causes the test to run for
                five seconds, wait for five seconds, and so on.
                This tests the locking primitive's ability to
                transition abruptly to and from idle.

locktorture.torture_runnable= [BOOT]
                Start locktorture running at boot time.

locktorture.torture_type= [KNL]
                Specify the locking implementation to test.

locktorture.verbose= [KNL]
                Enable additional printk() statements.

logibm.irq=     [HW,MOUSE] Logitech Bus Mouse Driver
                Format: <irq>

loglevel=       All Kernel Messages with a loglevel smaller than the
                console loglevel will be printed to the console. It can
                also be changed with klogd or other programs. The
                loglevels are defined as follows:

                0 (KERN_EMERG)          system is unusable
                1 (KERN_ALERT)          action must be taken immediately
                2 (KERN_CRIT)           critical conditions
                3 (KERN_ERR)            error conditions
                4 (KERN_WARNING)        warning conditions
                5 (KERN_NOTICE)         normal but significant condition
                6 (KERN_INFO)           informational
                7 (KERN_DEBUG)          debug-level messages

log_buf_len=n[KMG]      Sets the size of the printk ring buffer,
                in bytes.  n must be a power of two and greater
                than the minimal size. The minimal size is defined
                by LOG_BUF_SHIFT kernel config parameter. There is
                also CONFIG_LOG_CPU_MAX_BUF_SHIFT config parameter
                that allows to increase the default size depending on
                the number of CPUs. See init/Kconfig for more details.

logo.nologo     [FB] Disables display of the built-in Linux logo.
                This may be used to provide more screen space for
                kernel log messages and is useful when debugging
                kernel boot problems.

lp=0            [LP]    Specify parallel ports to use, e.g,
lp=port[,port...]       lp=none,parport0 (lp0 not configured, lp1 uses
lp=reset                first parallel port). `lp=0' disables the
lp=auto                 printer driver. `lp=reset' (which can be
                        specified in addition to the ports) causes
                        attached printers to be reset. Using
                        lp=port1,port2,... specifies the parallel ports
                        to associate lp devices with, starting with
                        lp0. A port specification may be `none' to skip
                        that lp device, or a parport name such as
                        `parport0'. Specifying `lp=auto' instead of a
                        port specification list means that device IDs
```

```
                        from each port should be examined, to see if
                        an IEEE 1284-compliant printer is attached; if
                        so, the driver will manage that printer.
                        See also header of drivers/char/lp.c.

lpj=n           [KNL]
                Sets loops_per_jiffy to given constant, thus avoiding
                time-consuming boot-time autodetection (up to 250 ms per
                CPU). 0 enables autodetection (default). To determine
                the correct value for your kernel, boot with normal
                autodetection and see what value is printed. Note that
                on SMP systems the preset will be applied to all CPUs,
                which is likely to cause problems if your CPUs need
                significantly divergent settings. An incorrect value
                will cause delays in the kernel to be wrong, leading to
                unpredictable I/O errors and other breakage. Although
                unlikely, in the extreme case this might damage your
                hardware.

ltpc=           [NET]
                Format: <io>,<irq>,<dma>

machvec=        [IA-64] Force the use of a particular machine-vector
                (machvec) in a generic kernel.
                Example: machvec=hpzx1_swiotlb

machtype=       [Loongson] Share the same kernel image file between different
                 yeeloong laptop.
                Example: machtype=lemote-yeeloong-2f-7inch

max_addr=nn[KMG]        [KNL,BOOT,ia64] All physical memory greater
                than or equal to this physical address is ignored.

maxcpus=        [SMP] Maximum number of processors that an SMP kernel
                will bring up during bootup.  maxcpus=n : n >= 0 limits
                the kernel to bring up `n' processors. Surely after
                bootup you can bring up the other plugged cpu by executing
                ``echo 1 > /sys/devices/system/cpu/cpuX/online''. So maxcpus
                only takes effect during system bootup.
                While n=0 is a special case, it is equivalent to ``nosmp'',
                which also disables the IO APIC.

max_loop=       [LOOP] The number of loop block devices that get
(loop.max_loop) unconditionally pre-created at init time. The default
                number is configured by BLK_DEV_LOOP_MIN_COUNT. Instead
                of statically allocating a predefined number, loop
                devices can be requested on-demand with the
                /dev/loop-control interface.

mce             [X86-32] Machine Check Exception

mce=option      [X86-64] See Documentation/x86/x86_64/boot-options.txt

md=             [HW] RAID subsystems devices and level
                See Documentation/admin-guide/md.rst.

mdacon=         [MDA]
                Format: <first>,<last>
```

```
                Specifies range of consoles to be captured by the MDA.

mem=nn[KMG]     [KNL,BOOT] Force usage of a specific amount of memory
                Amount of memory to be used when the kernel is not able
                to see the whole system memory or for test.
                [X86] Work as limiting max address. Use together
                with memmap= to avoid physical address space collisions.
                Without memmap= PCI devices could be placed at addresses
                belonging to unused RAM.

mem=nopentium   [BUGS=X86-32] Disable usage of 4MB pages for kernel
                memory.

memchunk=nn[KMG]
                [KNL,SH] Allow user to override the default size for
                per-device physically contiguous DMA buffers.

memhp_default_state=online/offline
                [KNL] Set the initial state for the memory hotplug
                onlining policy. If not specified, the default value is
                set according to the
                CONFIG_MEMORY_HOTPLUG_DEFAULT_ONLINE kernel config
                option.
                See Documentation/memory-hotplug.txt.

memmap=exactmap [KNL,X86] Enable setting of an exact
                E820 memory map, as specified by the user.
                Such memmap=exactmap lines can be constructed based on
                BIOS output or other requirements. See the memmap=nn@ss
                option description.

memmap=nn[KMG]@ss[KMG]
                [KNL] Force usage of a specific region of memory.
                Region of memory to be used is from ss to ss+nn.
                If @ss[KMG] is omitted, it is equivalent to mem=nn[KMG],
                which limits max address to nn[KMG].
                Multiple different regions can be specified,
                comma delimited.
                Example:
                        memmap=100M@2G,100M#3G,1G!1024G

memmap=nn[KMG]#ss[KMG]
                [KNL,ACPI] Mark specific memory as ACPI data.
                Region of memory to be marked is from ss to ss+nn.

memmap=nn[KMG]$ss[KMG]
                [KNL,ACPI] Mark specific memory as reserved.
                Region of memory to be reserved is from ss to ss+nn.
                Example: Exclude memory from 0x18690000-0x1869ffff
                         memmap=64K$0x18690000
                         or
                         memmap=0x10000$0x18690000
                Some bootloaders may need an escape character before `$',
                like Grub2, otherwise `$' and the following number
                will be eaten.

memmap=nn[KMG]!ss[KMG]
                [KNL,X86] Mark specific memory as protected.
```

```
                Region of memory to be used, from ss to ss+nn.
                The memory region may be marked as e820 type 12 (0xc)
                and is NVDIMM or ADR memory.

memory_corruption_check=0/1 [X86]
                Some BIOSes seem to corrupt the first 64k of
                memory when doing things like suspend/resume.
                Setting this option will scan the memory
                looking for corruption.  Enabling this will
                both detect corruption and prevent the kernel
                from using the memory being corrupted.
                However, its intended as a diagnostic tool; if
                repeatable BIOS-originated corruption always
                affects the same memory, you can use memmap=
                to prevent the kernel from using that memory.

memory_corruption_check_size=size [X86]
                By default it checks for corruption in the low
                64k, making this memory unavailable for normal
                use.  Use this parameter to scan for
                corruption in more or less memory.

memory_corruption_check_period=seconds [X86]
                By default it checks for corruption every 60
                seconds.  Use this parameter to check at some
                other rate.  0 disables periodic checking.

memtest=        [KNL,X86,ARM] Enable memtest
                Format: <integer>
                default : 0 <disable>
                Specifies the number of memtest passes to be
                performed. Each pass selects another test
                pattern from a given set of patterns. Memtest
                fills the memory with this pattern, validates
                memory contents and reserves bad memory
                regions that are detected.

mem_encrypt=    [X86-64] AMD Secure Memory Encryption (SME) control
                Valid arguments: on, off
                Default (depends on kernel configuration option):
                  on  (CONFIG_AMD_MEM_ENCRYPT_ACTIVE_BY_DEFAULT=y)
                  off (CONFIG_AMD_MEM_ENCRYPT_ACTIVE_BY_DEFAULT=n)
                mem_encrypt=on:         Activate SME
                mem_encrypt=off:        Do not activate SME

                Refer to Documentation/x86/amd-memory-encryption.txt
                for details on when memory encryption can be activated.

mem_sleep_default=      [SUSPEND] Default system suspend mode:
                s2idle  - Suspend-To-Idle
                shallow - Power-On Suspend or equivalent (if supported)
                deep    - Suspend-To-RAM or equivalent (if supported)
                See Documentation/power/states.txt.

meye.*=         [HW] Set MotionEye Camera parameters
                See Documentation/video4linux/meye.txt.

mfgpt_irq=      [IA-32] Specify the IRQ to use for the
```

```
                Multi-Function General Purpose Timers on AMD Geode
                platforms.

mfgptfix        [X86-32] Fix MFGPT timers on AMD Geode platforms when
                the BIOS has incorrectly applied a workaround. TinyBIOS
                version 0.98 is known to be affected, 0.99 fixes the
                problem by letting the user disable the workaround.

mga=            [HW,DRM]

min_addr=nn[KMG]        [KNL,BOOT,ia64] All physical memory below this
                physical address is ignored.

mini2440=       [ARM,HW,KNL]
                Format:[0..2][b][c][t]
                Default: ``0tb''
                MINI2440 configuration specification:
                0 - The attached screen is the 3.5'' TFT
                1 - The attached screen is the 7'' TFT
                2 - The VGA Shield is attached (1024x768)
                Leaving out the screen size parameter will not load
                the TFT driver, and the framebuffer will be left
                unconfigured.
                b - Enable backlight. The TFT backlight pin will be
                linked to the kernel VESA blanking code and a GPIO
                LED. This parameter is not necessary when using the
                VGA shield.
                c - Enable the s3c camera interface.
                t - Reserved for enabling touchscreen support. The
                touchscreen support is not enabled in the mainstream
                kernel as of 2.6.30, a preliminary port can be found
                in the ``bleeding edge'' mini2440 support kernel at
                http://repo.or.cz/w/linux-2.6/mini2440.git

mminit_loglevel=
                [KNL] When CONFIG_DEBUG_MEMORY_INIT is set, this
                parameter allows control of the logging verbosity for
                the additional memory initialisation checks. A value
                of 0 disables mminit logging and a level of 4 will
                log everything. Information is printed at KERN_DEBUG
                so loglevel=8 may also need to be specified.

module.sig_enforce
                [KNL] When CONFIG_MODULE_SIG is set, this means that
                modules without (valid) signatures will fail to load.
                Note that if CONFIG_MODULE_SIG_FORCE is set, that
                is always true, so this option does nothing.

module_blacklist=  [KNL] Do not load a comma-separated list of
                modules.  Useful for debugging problem modules.

mousedev.tap_time=
                [MOUSE] Maximum time between finger touching and
                leaving touchpad surface for touch to be considered
                a tap and be reported as a left button click (for
                touchpads working in absolute mode only).
                Format: <msecs>
mousedev.xres=  [MOUSE] Horizontal screen resolution, used for devices
```

```
                reporting absolute coordinates, such as tablets
mousedev.yres=  [MOUSE] Vertical screen resolution, used for devices
                reporting absolute coordinates, such as tablets

movablecore=nn[KMG]     [KNL,X86,IA-64,PPC] This parameter
                is similar to kernelcore except it specifies the
                amount of memory used for migratable allocations.
                If both kernelcore and movablecore is specified,
                then kernelcore will be at *least* the specified
                value but may be more. If movablecore on its own
                is specified, the administrator must be careful
                that the amount of memory usable for all allocations
                is not too small.

movable_node    [KNL] Boot-time switch to make hotplugable memory
                NUMA nodes to be movable. This means that the memory
                of such nodes will be usable only for movable
                allocations which rules out almost all kernel
                allocations. Use with caution!

MTD_Partition=  [MTD]
                Format: <name>,<region-number>,<size>,<offset>

MTD_Region=     [MTD] Format:
                <name>,<region-number>[,<base>,<size>,<buswidth>,<altbuswidth>]

mtdparts=       [MTD]
                See drivers/mtd/cmdlinepart.c.

multitce=off    [PPC]  This parameter disables the use of the pSeries
                firmware feature for updating multiple TCE entries
                at a time.

onenand.bdry=   [HW,MTD] Flex-OneNAND Boundary Configuration

                Format: [die0_boundary][,die0_lock][,die1_boundary][,die1_lock]

                boundary - index of last SLC block on Flex-OneNAND.
                           The remaining blocks are configured as MLC blocks.
                lock     - Configure if Flex-OneNAND boundary should be locked.
                           Once locked, the boundary cannot be changed.
                           1 indicates lock status, 0 indicates unlock status.

mtdset=         [ARM]
                ARM/S3C2412 JIVE boot control

                See arch/arm/mach-s3c2412/mach-jive.c

mtouchusb.raw_coordinates=
                [HW] Make the MicroTouch USB driver use raw coordinates
                (`y', default) or cooked coordinates (`n')

mtrr_chunk_size=nn[KMG] [X86]
                used for mtrr cleanup. It is largest continuous chunk
                that could hold holes aka. UC entries.

mtrr_gran_size=nn[KMG] [X86]
                Used for mtrr cleanup. It is granularity of mtrr block.
```

```
                Default is 1.
                Large value could prevent small alignment from
                using up MTRRs.

mtrr_spare_reg_nr=n [X86]
                Format: <integer>
                Range: 0,7 : spare reg number
                Default : 1
                Used for mtrr cleanup. It is spare mtrr entries number.
                Set to 2 or more if your graphical card needs more.

n2=             [NET] SDL Inc. RISCom/N2 synchronous serial card

netdev=         [NET] Network devices parameters
                Format: <irq>,<io>,<mem_start>,<mem_end>,<name>
                Note that mem_start is often overloaded to mean
                something different and driver-specific.
                This usage is only documented in each driver source
                file if at all.

nf_conntrack.acct=
                [NETFILTER] Enable connection tracking flow accounting
                0 to disable accounting
                1 to enable accounting
                Default value is 0.

nfsaddrs=       [NFS] Deprecated.  Use ip= instead.
                See Documentation/filesystems/nfs/nfsroot.txt.

nfsroot=        [NFS] nfs root filesystem for disk-less boxes.
                See Documentation/filesystems/nfs/nfsroot.txt.

nfsrootdebug    [NFS] enable nfsroot debugging messages.
                See Documentation/filesystems/nfs/nfsroot.txt.

nfs.callback_nr_threads=
                [NFSv4] set the total number of threads that the
                NFS client will assign to service NFSv4 callback
                requests.

nfs.callback_tcpport=
                [NFS] set the TCP port on which the NFSv4 callback
                channel should listen.

nfs.cache_getent=
                [NFS] sets the pathname to the program which is used
                to update the NFS client cache entries.

nfs.cache_getent_timeout=
                [NFS] sets the timeout after which an attempt to
                update a cache entry is deemed to have failed.

nfs.idmap_cache_timeout=
                [NFS] set the maximum lifetime for idmapper cache
                entries.

nfs.enable_ino64=
                [NFS] enable 64-bit inode numbers.
```

```
                If zero, the NFS client will fake up a 32-bit inode
                number for the readdir() and stat() syscalls instead
                of returning the full 64-bit number.
                The default is to return 64-bit inode numbers.

        nfs.max_session_cb_slots=
                [NFSv4.1] Sets the maximum number of session
                slots the client will assign to the callback
                channel. This determines the maximum number of
                callbacks the client will process in parallel for
                a particular server.

        nfs.max_session_slots=
                [NFSv4.1] Sets the maximum number of session slots
                the client will attempt to negotiate with the server.
                This limits the number of simultaneous RPC requests
                that the client can send to the NFSv4.1 server.
                Note that there is little point in setting this
                value higher than the max_tcp_slot_table_limit.

        nfs.nfs4_disable_idmapping=
                [NFSv4] When set to the default of `1', this option
                ensures that both the RPC level authentication
                scheme and the NFS level operations agree to use
                numeric uids/gids if the mount is using the
                `sec=sys' security flavour. In effect it is
                disabling idmapping, which can make migration from
                legacy NFSv2/v3 systems to NFSv4 easier.
                Servers that do not support this mode of operation
                will be autodetected by the client, and it will fall
                back to using the idmapper.
                To turn off this behaviour, set the value to `0'.
        nfs.nfs4_unique_id=
                [NFS4] Specify an additional fixed unique ident-
                ification string that NFSv4 clients can insert into
                their nfs_client_id4 string.  This is typically a
                UUID that is generated at system install time.

        nfs.send_implementation_id =
                [NFSv4.1] Send client implementation identification
                information in exchange_id requests.
                If zero, no implementation identification information
                will be sent.
                The default is to send the implementation identification
                information.

        nfs.recover_lost_locks =
                [NFSv4] Attempt to recover locks that were lost due
                to a lease timeout on the server. Please note that
                doing this risks data corruption, since there are
                no guarantees that the file will remain unchanged
                after the locks are lost.
                If you want to enable the kernel legacy behaviour of
                attempting to recover these locks, then set this
                parameter to `1'.
                The default parameter value of `0' causes the kernel
                not to attempt recovery of lost locks.
```

```
nfs4.layoutstats_timer =
                [NFSv4.2] Change the rate at which the kernel sends
                layoutstats to the pNFS metadata server.

                Setting this to value to 0 causes the kernel to use
                whatever value is the default set by the layout
                driver. A non-zero value sets the minimum interval
                in seconds between layoutstats transmissions.

nfsd.nfs4_disable_idmapping=
                [NFSv4] When set to the default of `1', the NFSv4
                server will return only numeric uids and gids to
                clients using auth_sys, and will accept numeric uids
                and gids from such clients.  This is intended to ease
                migration from NFSv2/v3.

nmi_debug=      [KNL,SH] Specify one or more actions to take
                when a NMI is triggered.
                Format: [state][,regs][,debounce][,die]

nmi_watchdog=   [KNL,BUGS=X86] Debugging features for SMP kernels
                Format: [panic,][nopanic,][num]
                Valid num: 0 or 1
                0 - turn hardlockup detector in nmi_watchdog off
                1 - turn hardlockup detector in nmi_watchdog on
                When panic is specified, panic when an NMI watchdog
                timeout occurs (or `nopanic' to override the opposite
                default). To disable both hard and soft lockup detectors,
                please see `nowatchdog'.
                This is useful when you use a panic=... timeout and
                need the box quickly up again.

netpoll.carrier_timeout=
                [NET] Specifies amount of time (in seconds) that
                netpoll should wait for a carrier. By default netpoll
                waits 4 seconds.

no387           [BUGS=X86-32] Tells the kernel to use the 387 maths
                emulation library even if a 387 maths coprocessor
                is present.

no_console_suspend
                [HW] Never suspend the console
                Disable suspending of consoles during suspend and
                hibernate operations.  Once disabled, debugging
                messages can reach various consoles while the rest
                of the system is being put to sleep (ie, while
                debugging driver suspend/resume hooks).  This may
                not work reliably with all consoles, but is known
                to work with serial and VGA consoles.
                To facilitate more flexible debugging, we also add
                console_suspend, a printk module parameter to control
                it. Users could use console_suspend (usually
                /sys/module/printk/parameters/console_suspend) to
                turn on/off it dynamically.

noaliencache    [MM, NUMA, SLAB] Disables the allocation of alien
                caches in the slab allocator.  Saves per-node memory,
```

```
                but will impact performance.

noalign         [KNL,ARM]

noapic          [SMP,APIC] Tells the kernel to not make use of any
                IOAPICs that may be present in the system.

noautogroup     Disable scheduler automatic task group creation.

nobats          [PPC] Do not use BATs for mapping kernel lowmem
                on ``Classic'' PPC cores.

nocache         [ARM]

noclflush       [BUGS=X86] Don't use the CLFLUSH instruction

nodelayacct     [KNL] Disable per-task delay accounting

nodsp           [SH] Disable hardware DSP at boot time.

noefi           Disable EFI runtime services support.

noexec          [IA-64]

noexec          [X86]
                On X86-32 available only on PAE configured kernels.
                noexec=on: enable non-executable mappings (default)
                noexec=off: disable non-executable mappings

nosmap          [X86]
                Disable SMAP (Supervisor Mode Access Prevention)
                even if it is supported by processor.

nosmep          [X86]
                Disable SMEP (Supervisor Mode Execution Prevention)
                even if it is supported by processor.

noexec32        [X86-64]
                This affects only 32-bit executables.
                noexec32=on: enable non-executable mappings (default)
                        read doesn't imply executable mappings
                noexec32=off: disable non-executable mappings
                        read implies executable mappings

nofpu           [MIPS,SH] Disable hardware FPU at boot time.

nofxsr          [BUGS=X86-32] Disables x86 floating point extended
                register save and restore. The kernel will only save
                legacy floating-point registers on task switch.

nohugeiomap     [KNL,x86] Disable kernel huge I/O mappings.

nosmt           [KNL,S390] Disable symmetric multithreading (SMT).
                Equivalent to smt=1.

noxsave         [BUGS=X86] Disables x86 extended register state save
                and restore using xsave. The kernel will fallback to
                enabling legacy floating-point and sse state.
```

```
noxsaveopt      [X86] Disables xsaveopt used in saving x86 extended
                register states. The kernel will fall back to use
                xsave to save the states. By using this parameter,
                performance of saving the states is degraded because
                xsave doesn't support modified optimization while
                xsaveopt supports it on xsaveopt enabled systems.

noxsaves        [X86] Disables xsaves and xrstors used in saving and
                restoring x86 extended register state in compacted
                form of xsave area. The kernel will fall back to use
                xsaveopt and xrstor to save and restore the states
                in standard form of xsave area. By using this
                parameter, xsave area per process might occupy more
                memory on xsaves enabled systems.

nohlt           [BUGS=ARM,SH] Tells the kernel that the sleep(SH) or
                wfi(ARM) instruction doesn't work correctly and not to
                use it. This is also useful when using JTAG debugger.

no_file_caps    Tells the kernel not to honor file capabilities.  The
                only way then for a file to be executed with privilege
                is to be setuid root or executed by root.

nohalt          [IA-64] Tells the kernel not to use the power saving
                function PAL_HALT_LIGHT when idle. This increases
                power-consumption. On the positive side, it reduces
                interrupt wake-up latency, which may improve performance
                in certain environments such as networked servers or
                real-time systems.

nohibernate     [HIBERNATION] Disable hibernation and resume.

nohz=           [KNL] Boottime enable/disable dynamic ticks
                Valid arguments: on, off
                Default: on

nohz_full=      [KNL,BOOT]
                The argument is a cpu list, as described above.
                In kernels built with CONFIG_NO_HZ_FULL=y, set
                the specified list of CPUs whose tick will be stopped
                whenever possible. The boot CPU will be forced outside
                the range to maintain the timekeeping.  Any CPUs
                in this list will have their RCU callbacks offloaded,
                just as if they had also been called out in the
                rcu_nocbs= boot parameter.

noiotrap        [SH] Disables trapped I/O port accesses.

noirqdebug      [X86-32] Disables the code which attempts to detect and
                disable unhandled interrupt sources.

no_timer_check  [X86,APIC] Disables the code which tests for
                broken timer IRQ sources.

noisapnp        [ISAPNP] Disables ISA PnP code.

noinitrd        [RAM] Tells the kernel not to load any configured
```

```
                initial RAM disk.

nointremap      [X86-64, Intel-IOMMU] Do not enable interrupt
                remapping.
                [Deprecated - use intremap=off]

nointroute      [IA-64]

noinvpcid       [X86] Disable the INVPCID cpu feature.

nojitter        [IA-64] Disables jitter checking for ITC timers.

no-kvmclock     [X86,KVM] Disable paravirtualized KVM clock driver

no-kvmapf       [X86,KVM] Disable paravirtualized asynchronous page
                fault handling.

no-vmw-sched-clock
                [X86,PV_OPS] Disable paravirtualized VMware scheduler
                clock and use the default one.

no-steal-acc    [X86,KVM] Disable paravirtualized steal time accounting.
                steal time is computed, but won't influence scheduler
                behaviour

nolapic         [X86-32,APIC] Do not enable or use the local APIC.

nolapic_timer   [X86-32,APIC] Do not use the local APIC timer.

noltlbs         [PPC] Do not use large page/tlb entries for kernel
                lowmem mapping on PPC40x and PPC8xx

nomca           [IA-64] Disable machine check abort handling

nomce           [X86-32] Disable Machine Check Exception

nomfgpt         [X86-32] Disable Multi-Function General Purpose
                Timer usage (for AMD Geode machines).

nonmi_ipi       [X86] Disable using NMI IPIs during panic/reboot to
                shutdown the other cpus.  Instead use the REBOOT_VECTOR
                irq.

nomodule        Disable module load

nopat           [X86] Disable PAT (page attribute table extension of
                pagetables) support.

nopcid          [X86-64] Disable the PCID cpu feature.

norandmaps      Don't use address space randomization.  Equivalent to
                echo 0 > /proc/sys/kernel/randomize_va_space

noreplace-paravirt      [X86,IA-64,PV_OPS] Don't patch paravirt_ops

noreplace-smp   [X86-32,SMP] Don't replace SMP instructions
                with UP alternatives
```

```
nordrand        [X86] Disable kernel use of the RDRAND and
                RDSEED instructions even if they are supported
                by the processor.  RDRAND and RDSEED are still
                available to user space applications.

noresume        [SWSUSP] Disables resume and restores original swap
                space.

no-scroll       [VGA] Disables scrollback.
                This is required for the Braillex ib80-piezo Braille
                reader made by F.H. Papenmeier (Germany).

nosbagart       [IA-64]

nosep           [BUGS=X86-32] Disables x86 SYSENTER/SYSEXIT support.

nosmp           [SMP] Tells an SMP kernel to act as a UP kernel,
                and disable the IO APIC.  legacy for ``maxcpus=0''.

nosoftlockup    [KNL] Disable the soft-lockup detector.

nosync          [HW,M68K] Disables sync negotiation for all devices.

notsc           [BUGS=X86-32] Disable Time Stamp Counter

nowatchdog      [KNL] Disable both lockup detectors, i.e.
                soft-lockup and NMI watchdog (hard-lockup).

nowb            [ARM]

nox2apic        [X86-64,APIC] Do not enable x2APIC mode.

cpu0_hotplug    [X86] Turn on CPU0 hotplug feature when
                CONFIG_BOOTPARAM_HOTPLUG_CPU0 is off.
                Some features depend on CPU0. Known dependencies are:
                1. Resume from suspend/hibernate depends on CPU0.
                Suspend/hibernate will fail if CPU0 is offline and you
                need to online CPU0 before suspend/hibernate.
                2. PIC interrupts also depend on CPU0. CPU0 can't be
                removed if a PIC interrupt is detected.
                It's said poweroff/reboot may depend on CPU0 on some
                machines although I haven't seen such issues so far
                after CPU0 is offline on a few tested machines.
                If the dependencies are under your control, you can
                turn on cpu0_hotplug.

nps_mtm_hs_ctr= [KNL,ARC]
                This parameter sets the maximum duration, in
                cycles, each HW thread of the CTOP can run
                without interruptions, before HW switches it.
                The actual maximum duration is 16 times this
                parameter's value.
                Format: integer between 1 and 255
                Default: 255

nptcg=          [IA-64] Override max number of concurrent global TLB
                purges which is reported from either PAL_VM_SUMMARY or
                SAL PALO.
```

```
nr_cpus=        [SMP] Maximum number of processors that an SMP kernel
                could support.  nr_cpus=n : n >= 1 limits the kernel to
                support `n' processors. It could be larger than the
                number of already plugged CPU during bootup, later in
                runtime you can physically add extra cpu until it reaches
                n. So during boot up some boot time memory for per-cpu
                variables need be pre-allocated for later physical cpu
                hot plugging.

nr_uarts=       [SERIAL] maximum number of UARTs to be registered.

numa_balancing= [KNL,X86] Enable or disable automatic NUMA balancing.
                Allowed values are enable and disable

numa_zonelist_order= [KNL, BOOT] Select zonelist order for NUMA.
                `node', `default' can be specified
                This can be set from sysctl after boot.
                See Documentation/sysctl/vm.txt for details.

ohci1394_dma=early      [HW] enable debugging via the ohci1394 driver.
                See Documentation/debugging-via-ohci1394.txt for more
                info.

olpc_ec_timeout= [OLPC] ms delay when issuing EC commands
                Rather than timing out after 20 ms if an EC
                command is not properly ACKed, override the length
                of the timeout.  We have interrupts disabled while
                waiting for the ACK, so if this is set too high
                interrupts *may* be lost!

omap_mux=       [OMAP] Override bootloader pin multiplexing.
                Format: <mux_mode0.mode_name=value>...
                For example, to override I2C bus2:
                omap_mux=i2c2_scl.i2c2_scl=0x100,i2c2_sda.i2c2_sda=0x100

oprofile.timer= [HW]
                Use timer interrupt instead of performance counters

oprofile.cpu_type=      Force an oprofile cpu type
                This might be useful if you have an older oprofile
                userland or if you want common events.
                Format: { arch_perfmon }
                arch_perfmon: [X86] Force use of architectural
                        perfmon on Intel CPUs instead of the
                        CPU specific event set.
                timer: [X86] Force use of architectural NMI
                        timer mode (see also oprofile.timer
                        for generic hr timer mode)

oops=panic      Always panic on oopses. Default is to just kill the
                process, but there is a small probability of
                deadlocking the machine.
                This will also cause panics on machine check exceptions.
                Useful together with panic=30 to trigger a reboot.

OSS             [HW,OSS]
                See Documentation/sound/oss/oss-parameters.txt
```

```
page_owner=     [KNL] Boot-time page_owner enabling option.
                Storage of the information about who allocated
                each page is disabled in default. With this switch,
                we can turn it on.
                on: enable the feature

page_poison=    [KNL] Boot-time parameter changing the state of
                poisoning on the buddy allocator.
                off: turn off poisoning
                on: turn on poisoning

panic=          [KNL] Kernel behaviour on panic: delay <timeout>
                timeout > 0: seconds before rebooting
                timeout = 0: wait forever
                timeout < 0: reboot immediately
                Format: <timeout>

panic_on_warn   panic() instead of WARN().  Useful to cause kdump
                on a WARN().

crash_kexec_post_notifiers
                Run kdump after running panic-notifiers and dumping
                kmsg. This only for the users who doubt kdump always
                succeeds in any situation.
                Note that this also increases risks of kdump failure,
                because some panic notifiers can make the crashed
                kernel more unstable.

parkbd.port=    [HW] Parallel port number the keyboard adapter is
                connected to, default is 0.
                Format: <parport#>
parkbd.mode=    [HW] Parallel port keyboard adapter mode of operation,
                0 for XT, 1 for AT (default is AT).
                Format: <mode>

parport=        [HW,PPT] Specify parallel ports. 0 disables.
                Format: { 0 | auto | 0xBBB[,IRQ[,DMA]] }
                Use `auto' to force the driver to use any
                IRQ/DMA settings detected (the default is to
                ignore detected IRQ/DMA settings because of
                possible conflicts). You can specify the base
                address, IRQ, and DMA settings; IRQ and DMA
                should be numbers, or `auto' (for using detected
                settings on that particular port), or `nofifo'
                (to avoid using a FIFO even if it is detected).
                Parallel ports are assigned in the order they
                are specified on the command line, starting
                with parport0.

parport_init_mode=      [HW,PPT]
                Configure VIA parallel port to operate in
                a specific mode. This is necessary on Pegasos
                computer where firmware has no options for setting
                up parallel port mode and sets it to spp.
                Currently this function knows 686a and 8231 chips.
                Format: [spp|ps2|epp|ecp|ecpepp]
```

```
        pause_on_oops=
                        Halt all CPUs after the first oops has been printed for
                        the specified number of seconds.  This is to be used if
                        your oopses keep scrolling off the screen.

        pcbit=          [HW,ISDN]

        pcd.            [PARIDE]
                        See header of drivers/block/paride/pcd.c.
                        See also Documentation/blockdev/paride.txt.

        pci=option[,option...]  [PCI] various PCI subsystem options:
                earlydump       [X86] dump PCI config space before the kernel
                                changes anything
                off             [X86] don't probe for the PCI bus
                bios            [X86-32] force use of PCI BIOS, don't access
                                the hardware directly. Use this if your machine
                                has a non-standard PCI host bridge.
                nobios          [X86-32] disallow use of PCI BIOS, only direct
                                hardware access methods are allowed. Use this
                                if you experience crashes upon bootup and you
                                suspect they are caused by the BIOS.
                conf1           [X86] Force use of PCI Configuration Access
                                Mechanism 1 (config address in IO port 0xCF8,
                                data in IO port 0xCFC, both 32-bit).
                conf2           [X86] Force use of PCI Configuration Access
                                Mechanism 2 (IO port 0xCF8 is an 8-bit port for
                                the function, IO port 0xCFA, also 8-bit, sets
                                bus number. The config space is then accessed
                                through ports 0xC000-0xCFFF).
                                See http://wiki.osdev.org/PCI for more info
                                on the configuration access mechanisms.
                noaer           [PCIE] If the PCIEAER kernel config parameter is
                                enabled, this kernel boot option can be used to
                                disable the use of PCIE advanced error reporting.
                nodomains       [PCI] Disable support for multiple PCI
                                root domains (aka PCI segments, in ACPI-speak).
                nommconf        [X86] Disable use of MMCONFIG for PCI
                                Configuration
                check_enable_amd_mmconf [X86] check for and enable
                                properly configured MMIO access to PCI
                                config space on AMD family 10h CPU
                nomsi           [MSI] 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.
                noioapicquirk   [APIC] Disable all boot interrupt quirks.
                                Safety option to keep boot IRQs enabled. This
                                should never be necessary.
                ioapicreroute   [APIC] Enable rerouting of boot IRQs to the
                                primary IO-APIC for bridges that cannot disable
                                boot IRQs. This fixes a source of spurious IRQs
                                when the system masks IRQs.
                noioapicreroute [APIC] Disable workaround that uses the
                                boot IRQ equivalent of an IRQ that connects to
                                a chipset where boot IRQs cannot be disabled.
                                The opposite of ioapicreroute.
                biosirq         [X86-32] Use PCI BIOS calls to get the interrupt
                                routing table. These calls are known to be buggy
```

```
                on several machines and they hang the machine
                when used, but on other computers it's 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             [X86] Assign address space to expansion ROMs.
                Use with caution as certain devices share
                address decoders between ROMs and other
                resources.
norom           [X86] Do not assign address space to
                expansion ROMs that do not already have
                BIOS assigned address ranges.
nobar           [X86] Do not assign address space to the
                BARs that weren't assigned by the BIOS.
irqmask=0xMMMM  [X86] 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=0xAAAAA        [X86] Specify the physical address
                of the PIRQ table (normally generated
                by the BIOS) if it is outside the
                F0000h-100000h range.
lastbus=N       [X86] Scan all buses thru 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   [X86] Always assign all PCI bus
                numbers ourselves, overriding
                whatever the firmware may have done.
usepirqmask     [X86] 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          [X86] Do not use ACPI for IRQ routing
                or for PCI scanning.
use_crs         [X86] Use PCI host bridge window information
                from ACPI.  On BIOSes from 2008 or later, this
                is enabled by default.  If you need to use this,
                please report a bug.
nocrs           [X86] Ignore PCI host bridge windows from ACPI.
                If you need to use this, please report a bug.
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.
skip_isa_align  [X86] do not align io start addr, so can
                handle more pci cards
noearly         [X86] Don't do any early type 1 scanning.
                This might help on some broken boards which
                machine check when some devices' config space
                is read. But various workarounds are disabled
                and some IOMMU drivers will not work.
bfsort          Sort PCI devices into breadth-first order.
                This sorting is done to get a device
                order compatible with older (<= 2.4) kernels.
```

```
        nobfsort        Don't sort PCI devices into breadth-first order.
        pcie_bus_tune_off       Disable PCIe MPS (Max Payload Size)
                        tuning and use the BIOS-configured MPS defaults.
        pcie_bus_safe   Set every device's MPS to the largest value
                        supported by all devices below the root complex.
        pcie_bus_perf   Set device MPS to the largest allowable MPS
                        based on its parent bus. Also set MRRS (Max
                        Read Request Size) to the largest supported
                        value (no larger than the MPS that the device
                        or bus can support) for best performance.
        pcie_bus_peer2peer      Set every device's MPS to 128B, which
                        every device is guaranteed to support. This
                        configuration allows peer-to-peer DMA between
                        any pair of devices, possibly at the cost of
                        reduced performance.  This also guarantees
                        that hot-added devices will work.
        cbiosize=nn[KMG]        The fixed amount of bus space which is
                        reserved for the CardBus bridge's IO window.
                        The default value is 256 bytes.
        cbmemsize=nn[KMG]       The fixed amount of bus space which is
                        reserved for the CardBus bridge's memory
                        window. The default value is 64 megabytes.
        resource_alignment=
                        Format:
                        [<order of align>@][<domain>:]<bus>:<slot>.<func>[; ...]
                        [<order of align>@]pci:<vendor>:<device>\
                                        [:<subvendor>:<subdevice>][; ...]
                        Specifies alignment and device to reassign
                        aligned memory resources.
                        If <order of align> is not specified,
                        PAGE_SIZE is used as alignment.
                        PCI-PCI bridge can be specified, if resource
                        windows need to be expanded.
                        To specify the alignment for several
                        instances of a device, the PCI vendor,
                        device, subvendor, and subdevice may be
                        specified, e.g., 4096@pci:8086:9c22:103c:198f
        ecrc=           Enable/disable PCIe ECRC (transaction layer
                        end-to-end CRC checking).
                        bios: Use BIOS/firmware settings. This is the
                        the default.
                        off: Turn ECRC off
                        on: Turn ECRC on.
        hpiosize=nn[KMG]        The fixed amount of bus space which is
                        reserved for hotplug bridge's IO window.
                        Default size is 256 bytes.
        hpmemsize=nn[KMG]       The fixed amount of bus space which is
                        reserved for hotplug bridge's memory window.
                        Default size is 2 megabytes.
        hpbussize=nn    The minimum amount of additional bus numbers
                        reserved for buses below a hotplug bridge.
                        Default is 1.
        realloc=        Enable/disable reallocating PCI bridge resources
                        if allocations done by BIOS are too small to
                        accommodate resources required by all child
                        devices.
                        off: Turn realloc off
                        on: Turn realloc on
```

```
        realloc         same as realloc=on
        noari           do not use PCIe ARI.
        pcie_scan_all   Scan all possible PCIe devices.  Otherwise we
                        only look for one device below a PCIe downstream
                        port.

pcie_aspm=      [PCIE] Forcibly enable or disable PCIe Active State Power
                Management.
        off     Disable ASPM.
        force   Enable ASPM even on devices that claim not to support it.
                WARNING: Forcing ASPM on may cause system lockups.

pcie_hp=        [PCIE] PCI Express Hotplug driver options:
        nomsi   Do not use MSI for PCI Express Native Hotplug (this
                makes all PCIe ports use INTx for hotplug services).

pcie_ports=     [PCIE] PCIe ports handling:
        auto    Ask the BIOS whether or not to use native PCIe services
                associated with PCIe ports (PME, hot-plug, AER).  Use
                them only if that is allowed by the BIOS.
        native  Use native PCIe services associated with PCIe ports
                unconditionally.
        compat  Treat PCIe ports as PCI-to-PCI bridges, disable the PCIe
                ports driver.

pcie_port_pm=   [PCIE] PCIe port power management handling:
        off     Disable power management of all PCIe ports
        force   Forcibly enable power management of all PCIe ports

pcie_pme=       [PCIE,PM] Native PCIe PME signaling options:
        nomsi   Do not use MSI for native PCIe PME signaling (this makes
                all PCIe root ports use INTx for all services).

pcmv=           [HW,PCMCIA] BadgePAD 4

pd_ignore_unused
                [PM]
                Keep all power-domains already enabled by bootloader on,
                even if no driver has claimed them. This is useful
                for debug and development, but should not be
                needed on a platform with proper driver support.

pd.             [PARIDE]
                See Documentation/blockdev/paride.txt.

pdcchassis=     [PARISC,HW] Disable/Enable PDC Chassis Status codes at
                boot time.
                Format: { 0 | 1 }
                See arch/parisc/kernel/pdc_chassis.c

percpu_alloc=   Select which percpu first chunk allocator to use.
                Currently supported values are ``embed'' and ``page''.
                Archs may support subset or none of the selections.
                See comments in mm/percpu.c for details on each
                allocator.  This parameter is primarily for debugging
                and performance comparison.

pf.             [PARIDE]
```

```
                See Documentation/blockdev/paride.txt.

pg.             [PARIDE]
                See Documentation/blockdev/paride.txt.

pirq=           [SMP,APIC] Manual mp-table setup
                See Documentation/x86/i386/IO-APIC.txt.

plip=           [PPT,NET] Parallel port network link
                Format: { parport<nr> | timid | 0 }
                See also Documentation/parport.txt.

pmtmr=          [X86] Manual setup of pmtmr I/O Port.
                Override pmtimer IOPort with a hex value.
                e.g. pmtmr=0x508

pnp.debug=1     [PNP]
                Enable PNP debug messages (depends on the
                CONFIG_PNP_DEBUG_MESSAGES option).  Change at run-time
                via /sys/module/pnp/parameters/debug.  We always show
                current resource usage; turning this on also shows
                possible settings and some assignment information.

pnpacpi=        [ACPI]
                { off }

pnpbios=        [ISAPNP]
                { on | off | curr | res | no-curr | no-res }

pnp_reserve_irq=
                [ISAPNP] Exclude IRQs for the autoconfiguration

pnp_reserve_dma=
                [ISAPNP] Exclude DMAs for the autoconfiguration

pnp_reserve_io= [ISAPNP] Exclude I/O ports for the autoconfiguration
                Ranges are in pairs (I/O port base and size).

pnp_reserve_mem=
                [ISAPNP] Exclude memory regions for the
                autoconfiguration.
                Ranges are in pairs (memory base and size).

ports=          [IP_VS_FTP] IPVS ftp helper module
                Default is 21.
                Up to 8 (IP_VS_APP_MAX_PORTS) ports
                may be specified.
                Format: <port>,<port>....

powersave=off   [PPC] This option disables power saving features.
                It specifically disables cpuidle and sets the
                platform machine description specific power_save
                function to NULL. On Idle the CPU just reduces
                execution priority.

ppc_strict_facility_enable
                [PPC] This option catches any kernel floating point,
                Altivec, VSX and SPE outside of regions specifically
```

```
                allowed (eg kernel_enable_fpu()/kernel_disable_fpu()).
                There is some performance impact when enabling this.

print-fatal-signals=
                [KNL] debug: print fatal signals

                If enabled, warn about various signal handling
                related application anomalies: too many signals,
                too many POSIX.1 timers, fatal signals causing a
                coredump - etc.

                If you hit the warning due to signal overflow,
                you might want to try ``ulimit -i unlimited''.

                default: off.

printk.always_kmsg_dump=
                Trigger kmsg_dump for cases other than kernel oops or
                panics
                Format: <bool>  (1/Y/y=enable, 0/N/n=disable)
                default: disabled

printk.devkmsg={on,off,ratelimit}
                Control writing to /dev/kmsg.
                on - unlimited logging to /dev/kmsg from userspace
                off - logging to /dev/kmsg disabled
                ratelimit - ratelimit the logging
                Default: ratelimit

printk.time=    Show timing data prefixed to each printk message line
                Format: <bool>  (1/Y/y=enable, 0/N/n=disable)

processor.max_cstate=   [HW,ACPI]
                Limit processor to maximum C-state
                max_cstate=9 overrides any DMI blacklist limit.

processor.nocst [HW,ACPI]
                Ignore the _CST method to determine C-states,
                instead using the legacy FADT method

profile=        [KNL] Enable kernel profiling via /proc/profile
                Format: [schedule,]<number>
                Param: ``schedule'' - profile schedule points.
                Param: <number> - step/bucket size as a power of 2 for
                        statistical time based profiling.
                Param: ``sleep'' - profile D-state sleeping (millisecs).
                        Requires CONFIG_SCHEDSTATS
                Param: ``kvm'' - profile VM exits.

prompt_ramdisk= [RAM] List of RAM disks to prompt for floppy disk
                before loading.
                See Documentation/blockdev/ramdisk.txt.

psmouse.proto=  [HW,MOUSE] Highest PS2 mouse protocol extension to
                probe for; one of (bare|imps|exps|lifebook|any).
psmouse.rate=   [HW,MOUSE] Set desired mouse report rate, in reports
                per second.
psmouse.resetafter=     [HW,MOUSE]
```

```
                Try to reset the device after so many bad packets
                (0 = never).
psmouse.resolution=
                [HW,MOUSE] Set desired mouse resolution, in dpi.
psmouse.smartscroll=
                [HW,MOUSE] Controls Logitech smartscroll autorepeat.
                0 = disabled, 1 = enabled (default).

pstore.backend= Specify the name of the pstore backend to use

pt.             [PARIDE]
                See Documentation/blockdev/paride.txt.

pty.legacy_count=
                [KNL] Number of legacy pty's. Overwrites compiled-in
                default number.

quiet           [KNL] Disable most log messages

r128=           [HW,DRM]

raid=           [HW,RAID]
                See Documentation/admin-guide/md.rst.

ramdisk_size=   [RAM] Sizes of RAM disks in kilobytes
                See Documentation/blockdev/ramdisk.txt.

ras=option[,option,...] [KNL] RAS-specific options

        cec_disable     [X86]
                        Disable the Correctable Errors Collector,
                        see CONFIG_RAS_CEC help text.

rcu_nocbs=      [KNL]
                The argument is a cpu list, as described above.

                In kernels built with CONFIG_RCU_NOCB_CPU=y, set
                the specified list of CPUs to be no-callback CPUs.
                Invocation of these CPUs' RCU callbacks will
                be offloaded to ``rcuox/N'' kthreads created for
                that purpose, where ``x'' is ``b'' for RCU-bh, ``p''
                for RCU-preempt, and ``s'' for RCU-sched, and ``N''
                is the CPU number.  This reduces OS jitter on the
                offloaded CPUs, which can be useful for HPC and
                real-time workloads.  It can also improve energy
                efficiency for asymmetric multiprocessors.

rcu_nocb_poll   [KNL]
                Rather than requiring that offloaded CPUs
                (specified by rcu_nocbs= above) explicitly
                awaken the corresponding ``rcuoN'' kthreads,
                make these kthreads poll for callbacks.
                This improves the real-time response for the
                offloaded CPUs by relieving them of the need to
                wake up the corresponding kthread, but degrades
                energy efficiency by requiring that the kthreads
                periodically wake up to do the polling.
```

```
rcutree.blimit= [KNL]
                Set maximum number of finished RCU callbacks to
                process in one batch.

rcutree.dump_tree=      [KNL]
                Dump the structure of the rcu_node combining tree
                out at early boot.  This is used for diagnostic
                purposes, to verify correct tree setup.

rcutree.gp_cleanup_delay=       [KNL]
                Set the number of jiffies to delay each step of
                RCU grace-period cleanup.

rcutree.gp_init_delay=  [KNL]
                Set the number of jiffies to delay each step of
                RCU grace-period initialization.

rcutree.gp_preinit_delay=       [KNL]
                Set the number of jiffies to delay each step of
                RCU grace-period pre-initialization, that is,
                the propagation of recent CPU-hotplug changes up
                the rcu_node combining tree.

rcutree.rcu_fanout_exact= [KNL]
                Disable autobalancing of the rcu_node combining
                tree.  This is used by rcutorture, and might
                possibly be useful for architectures having high
                cache-to-cache transfer latencies.

rcutree.rcu_fanout_leaf= [KNL]
                Change the number of CPUs assigned to each
                leaf rcu_node structure.  Useful for very
                large systems, which will choose the value 64,
                and for NUMA systems with large remote-access
                latencies, which will choose a value aligned
                with the appropriate hardware boundaries.

rcutree.jiffies_till_sched_qs= [KNL]
                Set required age in jiffies for a
                given grace period before RCU starts
                soliciting quiescent-state help from
                rcu_note_context_switch().

rcutree.jiffies_till_first_fqs= [KNL]
                Set delay from grace-period initialization to
                first attempt to force quiescent states.
                Units are jiffies, minimum value is zero,
                and maximum value is HZ.

rcutree.jiffies_till_next_fqs= [KNL]
                Set delay between subsequent attempts to force
                quiescent states.  Units are jiffies, minimum
                value is one, and maximum value is HZ.

rcutree.kthread_prio=    [KNL,BOOT]
                Set the SCHED_FIFO priority of the RCU per-CPU
                kthreads (rcuc/N). This value is also used for
                the priority of the RCU boost threads (rcub/N)
```

```
                and for the RCU grace-period kthreads (rcu_bh,
                rcu_preempt, and rcu_sched). If RCU_BOOST is
                set, valid values are 1-99 and the default is 1
                (the least-favored priority).  Otherwise, when
                RCU_BOOST is not set, valid values are 0-99 and
                the default is zero (non-realtime operation).

        rcutree.rcu_nocb_leader_stride= [KNL]
                Set the number of NOCB kthread groups, which
                defaults to the square root of the number of
                CPUs.  Larger numbers reduces the wakeup overhead
                on the per-CPU grace-period kthreads, but increases
                that same overhead on each group's leader.

        rcutree.qhimark= [KNL]
                Set threshold of queued RCU callbacks beyond which
                batch limiting is disabled.

        rcutree.qlowmark= [KNL]
                Set threshold of queued RCU callbacks below which
                batch limiting is re-enabled.

        rcutree.rcu_idle_gp_delay= [KNL]
                Set wakeup interval for idle CPUs that have
                RCU callbacks (RCU_FAST_NO_HZ=y).

        rcutree.rcu_idle_lazy_gp_delay= [KNL]
                Set wakeup interval for idle CPUs that have
                only ``lazy'' RCU callbacks (RCU_FAST_NO_HZ=y).
                Lazy RCU callbacks are those which RCU can
                prove do nothing more than free memory.

        rcutree.rcu_kick_kthreads= [KNL]
                Cause the grace-period kthread to get an extra
                wake_up() if it sleeps three times longer than
                it should at force-quiescent-state time.
                This wake_up() will be accompanied by a
                WARN_ONCE() splat and an ftrace_dump().

        rcuperf.gp_async= [KNL]
                Measure performance of asynchronous
                grace-period primitives such as call_rcu().

        rcuperf.gp_async_max= [KNL]
                Specify the maximum number of outstanding
                callbacks per writer thread.  When a writer
                thread exceeds this limit, it invokes the
                corresponding flavor of rcu_barrier() to allow
                previously posted callbacks to drain.

        rcuperf.gp_exp= [KNL]
                Measure performance of expedited synchronous
                grace-period primitives.

        rcuperf.holdoff= [KNL]
                Set test-start holdoff period.  The purpose of
                this parameter is to delay the start of the
                test until boot completes in order to avoid
```

```
                interference.

        rcuperf.nreaders= [KNL]
                Set number of RCU readers.  The value -1 selects
                N, where N is the number of CPUs.  A value
                ``n'' less than -1 selects N-n+1, where N is again
                the number of CPUs.  For example, -2 selects N
                (the number of CPUs), -3 selects N+1, and so on.
                A value of ``n'' less than or equal to -N selects
                a single reader.

        rcuperf.nwriters= [KNL]
                Set number of RCU writers.  The values operate
                the same as for rcuperf.nreaders.
                N, where N is the number of CPUs

        rcuperf.perf_runnable= [BOOT]
                Start rcuperf running at boot time.

        rcuperf.perf_type= [KNL]
                Specify the RCU implementation to test.

        rcuperf.shutdown= [KNL]
                Shut the system down after performance tests
                complete.  This is useful for hands-off automated
                testing.

        rcuperf.verbose= [KNL]
                Enable additional printk() statements.

        rcuperf.writer_holdoff= [KNL]
                Write-side holdoff between grace periods,
                in microseconds.  The default of zero says
                no holdoff.

        rcutorture.cbflood_inter_holdoff= [KNL]
                Set holdoff time (jiffies) between successive
                callback-flood tests.

        rcutorture.cbflood_intra_holdoff= [KNL]
                Set holdoff time (jiffies) between successive
                bursts of callbacks within a given callback-flood
                test.

        rcutorture.cbflood_n_burst= [KNL]
                Set the number of bursts making up a given
                callback-flood test.  Set this to zero to
                disable callback-flood testing.

        rcutorture.cbflood_n_per_burst= [KNL]
                Set the number of callbacks to be registered
                in a given burst of a callback-flood test.

        rcutorture.fqs_duration= [KNL]
                Set duration of force_quiescent_state bursts
                in microseconds.

        rcutorture.fqs_holdoff= [KNL]
```

```
                Set holdoff time within force_quiescent_state bursts
                in microseconds.

        rcutorture.fqs_stutter= [KNL]
                Set wait time between force_quiescent_state bursts
                in seconds.

        rcutorture.gp_cond= [KNL]
                Use conditional/asynchronous update-side
                primitives, if available.

        rcutorture.gp_exp= [KNL]
                Use expedited update-side primitives, if available.

        rcutorture.gp_normal= [KNL]
                Use normal (non-expedited) asynchronous
                update-side primitives, if available.

        rcutorture.gp_sync= [KNL]
                Use normal (non-expedited) synchronous
                update-side primitives, if available.  If all
                of rcutorture.gp_cond=, rcutorture.gp_exp=,
                rcutorture.gp_normal=, and rcutorture.gp_sync=
                are zero, rcutorture acts as if is interpreted
                they are all non-zero.

        rcutorture.n_barrier_cbs= [KNL]
                Set callbacks/threads for rcu_barrier() testing.

        rcutorture.nfakewriters= [KNL]
                Set number of concurrent RCU writers.  These just
                stress RCU, they don't participate in the actual
                test, hence the ``fake''.

        rcutorture.nreaders= [KNL]
                Set number of RCU readers.  The value -1 selects
                N-1, where N is the number of CPUs.  A value
                ``n'' less than -1 selects N-n-2, where N is again
                the number of CPUs.  For example, -2 selects N
                (the number of CPUs), -3 selects N+1, and so on.

        rcutorture.object_debug= [KNL]
                Enable debug-object double-call_rcu() testing.

        rcutorture.onoff_holdoff= [KNL]
                Set time (s) after boot for CPU-hotplug testing.

        rcutorture.onoff_interval= [KNL]
                Set time (s) between CPU-hotplug operations, or
                zero to disable CPU-hotplug testing.

        rcutorture.shuffle_interval= [KNL]
                Set task-shuffle interval (s).  Shuffling tasks
                allows some CPUs to go into dyntick-idle mode
                during the rcutorture test.

        rcutorture.shutdown_secs= [KNL]
                Set time (s) after boot system shutdown.  This
```

```
                is useful for hands-off automated testing.

rcutorture.stall_cpu= [KNL]
                Duration of CPU stall (s) to test RCU CPU stall
                warnings, zero to disable.

rcutorture.stall_cpu_holdoff= [KNL]
                Time to wait (s) after boot before inducing stall.

rcutorture.stat_interval= [KNL]
                Time (s) between statistics printk()s.

rcutorture.stutter= [KNL]
                Time (s) to stutter testing, for example, specifying
                five seconds causes the test to run for five seconds,
                wait for five seconds, and so on.  This tests RCU's
                ability to transition abruptly to and from idle.

rcutorture.test_boost= [KNL]
                Test RCU priority boosting?  0=no, 1=maybe, 2=yes.
                ``Maybe'' means test if the RCU implementation
                under test support RCU priority boosting.

rcutorture.test_boost_duration= [KNL]
                Duration (s) of each individual boost test.

rcutorture.test_boost_interval= [KNL]
                Interval (s) between each boost test.

rcutorture.test_no_idle_hz= [KNL]
                Test RCU's dyntick-idle handling.  See also the
                rcutorture.shuffle_interval parameter.

rcutorture.torture_runnable= [BOOT]
                Start rcutorture running at boot time.

rcutorture.torture_type= [KNL]
                Specify the RCU implementation to test.

rcutorture.verbose= [KNL]
                Enable additional printk() statements.

rcupdate.rcu_cpu_stall_suppress= [KNL]
                Suppress RCU CPU stall warning messages.

rcupdate.rcu_cpu_stall_timeout= [KNL]
                Set timeout for RCU CPU stall warning messages.

rcupdate.rcu_expedited= [KNL]
                Use expedited grace-period primitives, for
                example, synchronize_rcu_expedited() instead
                of synchronize_rcu().  This reduces latency,
                but can increase CPU utilization, degrade
                real-time latency, and degrade energy efficiency.
                No effect on CONFIG_TINY_RCU kernels.

rcupdate.rcu_normal= [KNL]
                Use only normal grace-period primitives,
```

```
                for example, synchronize_rcu() instead of
                synchronize_rcu_expedited().  This improves
                real-time latency, CPU utilization, and
                energy efficiency, but can expose users to
                increased grace-period latency.  This parameter
                overrides rcupdate.rcu_expedited.  No effect on
                CONFIG_TINY_RCU kernels.

        rcupdate.rcu_normal_after_boot= [KNL]
                Once boot has completed (that is, after
                rcu_end_inkernel_boot() has been invoked), use
                only normal grace-period primitives.  No effect
                on CONFIG_TINY_RCU kernels.

        rcupdate.rcu_task_stall_timeout= [KNL]
                Set timeout in jiffies for RCU task stall warning
                messages.  Disable with a value less than or equal
                to zero.

        rcupdate.rcu_self_test= [KNL]
                Run the RCU early boot self tests

        rcupdate.rcu_self_test_bh= [KNL]
                Run the RCU bh early boot self tests

        rcupdate.rcu_self_test_sched= [KNL]
                Run the RCU sched early boot self tests

        rdinit=         [KNL]
                Format: <full_path>
                Run specified binary instead of /init from the ramdisk,
                used for early userspace startup. See initrd.

        rdt=            [HW,X86,RDT]
                Turn on/off individual RDT features. List is:
                cmt, mbmtotal, mbmlocal, l3cat, l3cdp, l2cat, mba.
                E.g. to turn on cmt and turn off mba use:
                        rdt=cmt,!mba

        reboot=         [KNL]
                Format (x86 or x86_64):
                        [w[arm] | c[old] | h[ard] | s[oft] | g[pio]] \
                        [[,]s[mp]#### \
                        [[,]b[ios] | a[cpi] | k[bd] | t[riple] | e[fi] | p[ci]] \
                        [[,]f[orce]
                Where reboot_mode is one of warm (soft) or cold (hard) or gpio,
                      reboot_type is one of bios, acpi, kbd, triple, efi, or pci,
                      reboot_force is either force or not specified,
                      reboot_cpu is s[mp]#### with #### being the processor
                                to be used for rebooting.

        relax_domain_level=
                [KNL, SMP] Set scheduler's default relax_domain_level.
                See Documentation/cgroup-v1/cpusets.txt.

        reserve=        [KNL,BUGS] Force the kernel to ignore some iomem area

        reservetop=     [X86-32]
```

```
                Format: nn[KMG]
                Reserves a hole at the top of the kernel virtual
                address space.

reservelow=     [X86]
                Format: nn[K]
                Set the amount of memory to reserve for BIOS at
                the bottom of the address space.

reset_devices   [KNL] Force drivers to reset the underlying device
                during initialization.

resume=         [SWSUSP]
                Specify the partition device for software suspend
                Format:
                {/dev/<dev> | PARTUUID=<uuid> | <int>:<int> | <hex>}

resume_offset=  [SWSUSP]
                Specify the offset from the beginning of the partition
                given by ``resume='' at which the swap header is located,
                in <PAGE_SIZE> units (needed only for swap files).
                See  Documentation/power/swsusp-and-swap-files.txt

resumedelay=    [HIBERNATION] Delay (in seconds) to pause before attempting to
                read the resume files

resumewait      [HIBERNATION] Wait (indefinitely) for resume device to show up.
                Useful for devices that are detected asynchronously
                (e.g. USB and MMC devices).

hibernate=      [HIBERNATION]
        noresume        Don't check if there's a hibernation image
                        present during boot.
        nocompress      Don't compress/decompress hibernation images.
        no              Disable hibernation and resume.
        protect_image   Turn on image protection during restoration
                        (that will set all pages holding image data
                        during restoration read-only).

retain_initrd   [RAM] Keep initrd memory after extraction

rfkill.default_state=
        0       ``airplane mode''.  All wifi, bluetooth, wimax, gps, fm,
                etc. communication is blocked by default.
        1       Unblocked.

rfkill.master_switch_mode=
        0       The ``airplane mode'' button does nothing.
        1       The ``airplane mode'' button toggles between everything
                blocked and the previous configuration.
        2       The ``airplane mode'' button toggles between everything
                blocked and everything unblocked.

rhash_entries=  [KNL,NET]
                Set number of hash buckets for route cache

ring3mwait=disable
                [KNL] Disable ring 3 MONITOR/MWAIT feature on supported
```

```
                CPUs.

ro              [KNL] Mount root device read-only on boot

rodata=         [KNL]
        on      Mark read-only kernel memory as read-only (default).
        off     Leave read-only kernel memory writable for debugging.

rockchip.usb_uart
                Enable the uart passthrough on the designated usb port
                on Rockchip SoCs. When active, the signals of the
                debug-uart get routed to the D+ and D- pins of the usb
                port and the regular usb controller gets disabled.

root=           [KNL] Root filesystem
                See name_to_dev_t comment in init/do_mounts.c.

rootdelay=      [KNL] Delay (in seconds) to pause before attempting to
                mount the root filesystem

rootflags=      [KNL] Set root filesystem mount option string

rootfstype=     [KNL] Set root filesystem type

rootwait        [KNL] Wait (indefinitely) for root device to show up.
                Useful for devices that are detected asynchronously
                (e.g. USB and MMC devices).

rproc_mem=nn[KMG][@address]
                [KNL,ARM,CMA] Remoteproc physical memory block.
                Memory area to be used by remote processor image,
                managed by CMA.

rw              [KNL] Mount root device read-write on boot

S               [KNL] Run init in single mode

s390_iommu=     [HW,S390]
                Set s390 IOTLB flushing mode
        strict
                With strict flushing every unmap operation will result in
                an IOTLB flush. Default is lazy flushing before reuse,
                which is faster.

sa1100ir        [NET]
                See drivers/net/irda/sa1100_ir.c.

sbni=           [NET] Granch SBNI12 leased line adapter

sched_debug     [KNL] Enables verbose scheduler debug messages.

schedstats=     [KNL,X86] Enable or disable scheduled statistics.
                Allowed values are enable and disable. This feature
                incurs a small amount of overhead in the scheduler
                but is useful for debugging and performance tuning.

skew_tick=      [KNL] Offset the periodic timer tick per cpu to mitigate
                xtime_lock contention on larger systems, and/or RCU lock
```

```
                contention on all systems with CONFIG_MAXSMP set.
                Format: { ``0'' | ``1'' }
                0 -- disable. (may be 1 via CONFIG_CMDLINE=''skew_tick=1''
                1 -- enable.
                Note: increases power consumption, thus should only be
                enabled if running jitter sensitive (HPC/RT) workloads.

security=       [SECURITY] Choose a security module to enable at boot.
                If this boot parameter is not specified, only the first
                security module asking for security registration will be
                loaded. An invalid security module name will be treated
                as if no module has been chosen.

selinux=        [SELINUX] Disable or enable SELinux at boot time.
                Format: { ``0'' | ``1'' }
                See security/selinux/Kconfig help text.
                0 -- disable.
                1 -- enable.
                Default value is set via kernel config option.
                If enabled at boot time, /selinux/disable can be used
                later to disable prior to initial policy load.

apparmor=       [APPARMOR] Disable or enable AppArmor at boot time
                Format: { ``0'' | ``1'' }
                See security/apparmor/Kconfig help text
                0 -- disable.
                1 -- enable.
                Default value is set via kernel config option.

serialnumber    [BUGS=X86-32]

shapers=        [NET]
                Maximal number of shapers.

simeth=         [IA-64]
simscsi=

slram=          [HW,MTD]

slab_nomerge    [MM]
                Disable merging of slabs with similar size. May be
                necessary if there is some reason to distinguish
                allocs to different slabs, especially in hardened
                environments where the risk of heap overflows and
                layout control by attackers can usually be
                frustrated by disabling merging. This will reduce
                most of the exposure of a heap attack to a single
                cache (risks via metadata attacks are mostly
                unchanged). Debug options disable merging on their
                own.
                For more information see Documentation/vm/slub.txt.

slab_max_order= [MM, SLAB]
                Determines the maximum allowed order for slabs.
                A high setting may cause OOMs due to memory
                fragmentation.  Defaults to 1 for systems with
                more than 32MB of RAM, 0 otherwise.
```

```
        slub_debug[=options[,slabs]]    [MM, SLUB]
                        Enabling slub_debug allows one to determine the
                        culprit if slab objects become corrupted. Enabling
                        slub_debug can create guard zones around objects and
                        may poison objects when not in use. Also tracks the
                        last alloc / free. For more information see
                        Documentation/vm/slub.txt.

        slub_memcg_sysfs=       [MM, SLUB]
                        Determines whether to enable sysfs directories for
                        memory cgroup sub-caches. 1 to enable, 0 to disable.
                        The default is determined by CONFIG_SLUB_MEMCG_SYSFS_ON.
                        Enabling this can lead to a very high number of debug
                        directories and files being created under
                        /sys/kernel/slub.

        slub_max_order= [MM, SLUB]
                        Determines the maximum allowed order for slabs.
                        A high setting may cause OOMs due to memory
                        fragmentation. For more information see
                        Documentation/vm/slub.txt.

        slub_min_objects=       [MM, SLUB]
                        The minimum number of objects per slab. SLUB will
                        increase the slab order up to slub_max_order to
                        generate a sufficiently large slab able to contain
                        the number of objects indicated. The higher the number
                        of objects the smaller the overhead of tracking slabs
                        and the less frequently locks need to be acquired.
                        For more information see Documentation/vm/slub.txt.

        slub_min_order= [MM, SLUB]
                        Determines the minimum page order for slabs. Must be
                        lower than slub_max_order.
                        For more information see Documentation/vm/slub.txt.

        slub_nomerge    [MM, SLUB]
                        Same with slab_nomerge. This is supported for legacy.
                        See slab_nomerge for more information.

        smart2=         [HW]
                        Format: <io1>[,<io2>[,...,<io8>]]

        smsc-ircc2.nopnp        [HW] Don't use PNP to discover SMC devices
        smsc-ircc2.ircc_cfg=    [HW] Device configuration I/O port
        smsc-ircc2.ircc_sir=    [HW] SIR base I/O port
        smsc-ircc2.ircc_fir=    [HW] FIR base I/O port
        smsc-ircc2.ircc_irq=    [HW] IRQ line
        smsc-ircc2.ircc_dma=    [HW] DMA channel
        smsc-ircc2.ircc_transceiver= [HW] Transceiver type:
                                0: Toshiba Satellite 1800 (GP data pin select)
                                1: Fast pin select (default)
                                2: ATC IRMode

        smt             [KNL,S390] Set the maximum number of threads (logical
                        CPUs) to use per physical CPU on systems capable of
                        symmetric multithreading (SMT). Will be capped to the
                        actual hardware limit.
```

```
                Format: <integer>
                Default: -1 (no limit)

softlockup_panic=
                [KNL] Should the soft-lockup detector generate panics.
                Format: <integer>

softlockup_all_cpu_backtrace=
                [KNL] Should the soft-lockup detector generate
                backtraces on all cpus.
                Format: <integer>

sonypi.*=       [HW] Sony Programmable I/O Control Device driver
                See Documentation/laptops/sonypi.txt

spia_io_base=   [HW,MTD]
spia_fio_base=
spia_pedr=
spia_peddr=

srcutree.counter_wrap_check [KNL]
                Specifies how frequently to check for
                grace-period sequence counter wrap for the
                srcu_data structure's ->srcu_gp_seq_needed field.
                The greater the number of bits set in this kernel
                parameter, the less frequently counter wrap will
                be checked for.  Note that the bottom two bits
                are ignored.

srcutree.exp_holdoff [KNL]
                Specifies how many nanoseconds must elapse
                since the end of the last SRCU grace period for
                a given srcu_struct until the next normal SRCU
                grace period will be considered for automatic
                expediting.  Set to zero to disable automatic
                expediting.

stack_guard_gap=        [MM]
                override the default stack gap protection. The value
                is in page units and it defines how many pages prior
                to (for stacks growing down) resp. after (for stacks
                growing up) the main stack are reserved for no other
                mapping. Default value is 256 pages.

stacktrace      [FTRACE]
                Enabled the stack tracer on boot up.

stacktrace_filter=[function-list]
                [FTRACE] Limit the functions that the stack tracer
                will trace at boot up. function-list is a comma separated
                list of functions. This list can be changed at run
                time by the stack_trace_filter file in the debugfs
                tracing directory. Note, this enables stack tracing
                and the stacktrace above is not needed.

sti=            [PARISC,HW]
                Format: <num>
                Set the STI (builtin display/keyboard on the HP-PARISC
```

```
                machines) console (graphic card) which should be used
                as the initial boot-console.
                See also comment in drivers/video/console/sticore.c.

        sti_font=       [HW]
                See comment in drivers/video/console/sticore.c.

        stifb=          [HW]
                Format: bpp:<bpp1>[:<bpp2>[:<bpp3>...]]

        sunrpc.min_resvport=
        sunrpc.max_resvport=
                [NFS,SUNRPC]
                SunRPC servers often require that client requests
                originate from a privileged port (i.e. a port in the
                range 0 < portnr < 1024).
                An administrator who wishes to reserve some of these
                ports for other uses may adjust the range that the
                kernel's sunrpc client considers to be privileged
                using these two parameters to set the minimum and
                maximum port values.

        sunrpc.svc_rpc_per_connection_limit=
                [NFS,SUNRPC]
                Limit the number of requests that the server will
                process in parallel from a single connection.
                The default value is 0 (no limit).

        sunrpc.pool_mode=
                [NFS]
                Control how the NFS server code allocates CPUs to
                service thread pools.  Depending on how many NICs
                you have and where their interrupts are bound, this
                option will affect which CPUs will do NFS serving.
                Note: this parameter cannot be changed while the
                NFS server is running.

                auto        the server chooses an appropriate mode
                            automatically using heuristics
                global      a single global pool contains all CPUs
                percpu      one pool for each CPU
                pernode     one pool for each NUMA node (equivalent
                            to global on non-NUMA machines)

        sunrpc.tcp_slot_table_entries=
        sunrpc.udp_slot_table_entries=
                [NFS,SUNRPC]
                Sets the upper limit on the number of simultaneous
                RPC calls that can be sent from the client to a
                server. Increasing these values may allow you to
                improve throughput, but will also increase the
                amount of memory reserved for use by the client.

        suspend.pm_test_delay=
                [SUSPEND]
                Sets the number of seconds to remain in a suspend test
                mode before resuming the system (see
                /sys/power/pm_test). Only available when CONFIG_PM_DEBUG
```

```
                is set. Default value is 5.

swapaccount=[0|1]
                [KNL] Enable accounting of swap in memory resource
                controller if no parameter or 1 is given or disable
                it if 0 is given (See Documentation/cgroup-v1/memory.txt)

swiotlb=        [ARM,IA-64,PPC,MIPS,X86]
                Format: { <int> | force | noforce }
                <int> -- Number of I/O TLB slabs
                force -- force using of bounce buffers even if they
                         wouldn't be automatically used by the kernel
                noforce -- Never use bounce buffers (for debugging)

switches=       [HW,M68k]

sysfs.deprecated=0|1 [KNL]
                Enable/disable old style sysfs layout for old udev
                on older distributions. When this option is enabled
                very new udev will not work anymore. When this option
                is disabled (or CONFIG_SYSFS_DEPRECATED not compiled)
                in older udev will not work anymore.
                Default depends on CONFIG_SYSFS_DEPRECATED_V2 set in
                the kernel configuration.

sysrq_always_enabled
                [KNL]
                Ignore sysrq setting - this boot parameter will
                neutralize any effect of /proc/sys/kernel/sysrq.
                Useful for debugging.

tcpmhash_entries= [KNL,NET]
                Set the number of tcp_metrics_hash slots.
                Default value is 8192 or 16384 depending on total
                ram pages. This is used to specify the TCP metrics
                cache size. See Documentation/networking/ip-sysctl.txt
                ``tcp_no_metrics_save'' section for more details.

tdfx=           [HW,DRM]

test_suspend=   [SUSPEND][,N]
                Specify ``mem'' (for Suspend-to-RAM) or ``standby'' (for
                standby suspend) or ``freeze'' (for suspend type freeze)
                as the system sleep state during system startup with
                the optional capability to repeat N number of times.
                The system is woken from this state using a
                wakeup-capable RTC alarm.

thash_entries=  [KNL,NET]
                Set number of hash buckets for TCP connection

thermal.act=    [HW,ACPI]
                -1: disable all active trip points in all thermal zones
                <degrees C>: override all lowest active trip points

thermal.crt=    [HW,ACPI]
                -1: disable all critical trip points in all thermal zones
                <degrees C>: override all critical trip points
```

```
thermal.nocrt=  [HW,ACPI]
                Set to disable actions on ACPI thermal zone
                critical and hot trip points.

thermal.off=    [HW,ACPI]
                1: disable ACPI thermal control

thermal.psv=    [HW,ACPI]
                -1: disable all passive trip points
                <degrees C>: override all passive trip points to this
                value

thermal.tzp=    [HW,ACPI]
                Specify global default ACPI thermal zone polling rate
                <deci-seconds>: poll all this frequency
                0: no polling (default)

threadirqs      [KNL]
                Force threading of all interrupt handlers except those
                marked explicitly IRQF_NO_THREAD.

tmem            [KNL,XEN]
                Enable the Transcendent memory driver if built-in.

tmem.cleancache=0|1 [KNL, XEN]
                Default is on (1). Disable the usage of the cleancache
                API to send anonymous pages to the hypervisor.

tmem.frontswap=0|1 [KNL, XEN]
                Default is on (1). Disable the usage of the frontswap
                API to send swap pages to the hypervisor. If disabled
                the selfballooning and selfshrinking are force disabled.

tmem.selfballooning=0|1 [KNL, XEN]
                Default is on (1). Disable the driving of swap pages
                to the hypervisor.

tmem.selfshrinking=0|1 [KNL, XEN]
                Default is on (1). Partial swapoff that immediately
                transfers pages from Xen hypervisor back to the
                kernel based on different criteria.

topology=       [S390]
                Format: {off | on}
                Specify if the kernel should make use of the cpu
                topology information if the hardware supports this.
                The scheduler will make use of this information and
                e.g. base its process migration decisions on it.
                Default is on.

topology_updates= [KNL, PPC, NUMA]
                Format: {off}
                Specify if the kernel should ignore (off)
                topology updates sent by the hypervisor to this
                LPAR.

tp720=          [HW,PS2]
```

```
tpm_suspend_pcr=[HW,TPM]
                Format: integer pcr id
                Specify that at suspend time, the tpm driver
                should extend the specified pcr with zeros,
                as a workaround for some chips which fail to
                flush the last written pcr on TPM_SaveState.
                This will guarantee that all the other pcrs
                are saved.

trace_buf_size=nn[KMG]
                [FTRACE] will set tracing buffer size on each cpu.

trace_event=[event-list]
                [FTRACE] Set and start specified trace events in order
                to facilitate early boot debugging. The event-list is a
                comma separated list of trace events to enable. See
                also Documentation/trace/events.txt

trace_options=[option-list]
                [FTRACE] Enable or disable tracer options at boot.
                The option-list is a comma delimited list of options
                that can be enabled or disabled just as if you were
                to echo the option name into

                    /sys/kernel/debug/tracing/trace_options

                For example, to enable stacktrace option (to dump the
                stack trace of each event), add to the command line:

                      trace_options=stacktrace

                See also Documentation/trace/ftrace.txt ``trace options''
                section.

tp_printk[FTRACE]
                Have the tracepoints sent to printk as well as the
                tracing ring buffer. This is useful for early boot up
                where the system hangs or reboots and does not give the
                option for reading the tracing buffer or performing a
                ftrace_dump_on_oops.

                To turn off having tracepoints sent to printk,
                 echo 0 > /proc/sys/kernel/tracepoint_printk
                Note, echoing 1 into this file without the
                tracepoint_printk kernel cmdline option has no effect.

                ** CAUTION **

                Having tracepoints sent to printk() and activating high
                frequency tracepoints such as irq or sched, can cause
                the system to live lock.

traceoff_on_warning
                [FTRACE] enable this option to disable tracing when a
                warning is hit. This turns off ``tracing_on''. Tracing can
                be enabled again by echoing `1' into the ``tracing_on''
                file located in /sys/kernel/debug/tracing/
```

```
                This option is useful, as it disables the trace before
                the WARNING dump is called, which prevents the trace to
                be filled with content caused by the warning output.

                This option can also be set at run time via the sysctl
                option:  kernel/traceoff_on_warning

transparent_hugepage=
                [KNL]
                Format: [always|madvise|never]
                Can be used to control the default behavior of the system
                with respect to transparent hugepages.
                See Documentation/vm/transhuge.txt for more details.

tsc=            Disable clocksource stability checks for TSC.
                Format: <string>
                [x86] reliable: mark tsc clocksource as reliable, this
                disables clocksource verification at runtime, as well
                as the stability checks done at bootup. Used to enable
                high-resolution timer mode on older hardware, and in
                virtualized environment.
                [x86] noirqtime: Do not use TSC to do irq accounting.
                Used to run time disable IRQ_TIME_ACCOUNTING on any
                platforms where RDTSC is slow and this accounting
                can add overhead.

turbografx.map[2|3]=    [HW,JOY]
                TurboGraFX parallel port interface
                Format:
                <port#>,<js1>,<js2>,<js3>,<js4>,<js5>,<js6>,<js7>
                See also Documentation/input/joystick-parport.txt

udbg-immortal   [PPC] When debugging early kernel crashes that
                happen after console_init() and before a proper
                console driver takes over, this boot options might
                help ``seeing'' what's going on.

uhash_entries=  [KNL,NET]
                Set number of hash buckets for UDP/UDP-Lite connections

uhci-hcd.ignore_oc=
                [USB] Ignore overcurrent events (default N).
                Some badly-designed motherboards generate lots of
                bogus events, for ports that aren't wired to
                anything.  Set this parameter to avoid log spamming.
                Note that genuine overcurrent events won't be
                reported either.

unknown_nmi_panic
                [X86] Cause panic on unknown NMI.

usbcore.authorized_default=
                [USB] Default USB device authorization:
                (default -1 = authorized except for wireless USB,
                0 = not authorized, 1 = authorized)

usbcore.autosuspend=
```

```
                [USB] The autosuspend time delay (in seconds) used
                for newly-detected USB devices (default 2).  This
                is the time required before an idle device will be
                autosuspended.  Devices for which the delay is set
                to a negative value won't be autosuspended at all.

usbcore.usbfs_snoop=
                [USB] Set to log all usbfs traffic (default 0 = off).

usbcore.usbfs_snoop_max=
                [USB] Maximum number of bytes to snoop in each URB
                (default = 65536).

usbcore.blinkenlights=
                [USB] Set to cycle leds on hubs (default 0 = off).

usbcore.old_scheme_first=
                [USB] Start with the old device initialization
                scheme (default 0 = off).

usbcore.usbfs_memory_mb=
                [USB] Memory limit (in MB) for buffers allocated by
                usbfs (default = 16, 0 = max = 2047).

usbcore.use_both_schemes=
                [USB] Try the other device initialization scheme
                if the first one fails (default 1 = enabled).

usbcore.initial_descriptor_timeout=
                [USB] Specifies timeout for the initial 64-byte
                USB_REQ_GET_DESCRIPTOR request in milliseconds
                (default 5000 = 5.0 seconds).

usbcore.nousb   [USB] Disable the USB subsystem

usbhid.mousepoll=
                [USBHID] The interval which mice are to be polled at.

usbhid.jspoll=
                [USBHID] The interval which joysticks are to be polled at.

usb-storage.delay_use=
                [UMS] The delay in seconds before a new device is
                scanned for Logical Units (default 1).

usb-storage.quirks=
                [UMS] A list of quirks entries to supplement or
                override the built-in unusual_devs list.  List
                entries are separated by commas.  Each entry has
                the form VID:PID:Flags where VID and PID are Vendor
                and Product ID values (4-digit hex numbers) and
                Flags is a set of characters, each corresponding
                to a common usb-storage quirk flag as follows:
                        a = SANE_SENSE (collect more than 18 bytes
                                of sense data);
                        b = BAD_SENSE (don't collect more than 18
                                bytes of sense data);
                        c = FIX_CAPACITY (decrease the reported
```

```
                                device capacity by one sector);
                        d = NO_READ_DISC_INFO (don't use
                                READ_DISC_INFO command);
                        e = NO_READ_CAPACITY_16 (don't use
                                READ_CAPACITY_16 command);
                        f = NO_REPORT_OPCODES (don't use report opcodes
                                command, uas only);
                        g = MAX_SECTORS_240 (don't transfer more than
                                240 sectors at a time, uas only);
                        h = CAPACITY_HEURISTICS (decrease the
                                reported device capacity by one
                                sector if the number is odd);
                        i = IGNORE_DEVICE (don't bind to this
                                device);
                        j = NO_REPORT_LUNS (don't use report luns
                                command, uas only);
                        l = NOT_LOCKABLE (don't try to lock and
                                unlock ejectable media);
                        m = MAX_SECTORS_64 (don't transfer more
                                than 64 sectors = 32 KB at a time);
                        n = INITIAL_READ10 (force a retry of the
                                initial READ(10) command);
                        o = CAPACITY_OK (accept the capacity
                                reported by the device);
                        p = WRITE_CACHE (the device cache is ON
                                by default);
                        r = IGNORE_RESIDUE (the device reports
                                bogus residue values);
                        s = SINGLE_LUN (the device has only one
                                Logical Unit);
                        t = NO_ATA_1X (don't allow ATA(12) and ATA(16)
                                commands, uas only);
                        u = IGNORE_UAS (don't bind to the uas driver);
                        w = NO_WP_DETECT (don't test whether the
                                medium is write-protected).
                        y = ALWAYS_SYNC (issue a SYNCHRONIZE_CACHE
                                even if the device claims no cache)
                Example: quirks=0419:aaf5:rl,0421:0433:rc

user_debug=     [KNL,ARM]
                Format: <int>
                See arch/arm/Kconfig.debug help text.
                         1 - undefined instruction events
                         2 - system calls
                         4 - invalid data aborts
                         8 - SIGSEGV faults
                        16 - SIGBUS faults
                Example: user_debug=31

userpte=
                [X86] Flags controlling user PTE allocations.

                        nohigh = do not allocate PTE pages in
                                HIGHMEM regardless of setting
                                of CONFIG_HIGHPTE.

vdso=           [X86,SH]
                On X86_32, this is an alias for vdso32=.  Otherwise:
```

```
                vdso=1: enable VDSO (the default)
                vdso=0: disable VDSO mapping

    vdso32=     [X86] Control the 32-bit vDSO
                vdso32=1: enable 32-bit VDSO
                vdso32=0 or vdso32=2: disable 32-bit VDSO

                See the help text for CONFIG_COMPAT_VDSO for more
                details.  If CONFIG_COMPAT_VDSO is set, the default is
                vdso32=0; otherwise, the default is vdso32=1.

                For compatibility with older kernels, vdso32=2 is an
                alias for vdso32=0.

                Try vdso32=0 if you encounter an error that says:
                dl_main: Assertion `(void *) ph->p_vaddr == _rtld_local._dl_sysinfo_ds

    vector=     [IA-64,SMP]
                vector=percpu: enable percpu vector domain

    video=      [FB] Frame buffer configuration
                See Documentation/fb/modedb.txt.

    video.brightness_switch_enabled= [0,1]
                If set to 1, on receiving an ACPI notify event
                generated by hotkey, video driver will adjust brightness
                level and then send out the event to user space through
                the allocated input device; If set to 0, video driver
                will only send out the event without touching backlight
                brightness level.
                default: 1

    virtio_mmio.device=
                [VMMIO] Memory mapped virtio (platform) device.

                        <size>@<baseaddr>:<irq>[:<id>]
                where:
                        <size>     := size (can use standard suffixes
                                        like K, M and G)
                        <baseaddr> := physical base address
                        <irq>      := interrupt number (as passed to
                                        request_irq())
                        <id>       := (optional) platform device id
                example:
                        virtio_mmio.device=1K@0x100b0000:48:7

                Can be used multiple times for multiple devices.

    vga=        [BOOT,X86-32] Select a particular video mode
                See Documentation/x86/boot.txt and
                Documentation/svga.txt.
                Use vga=ask for menu.
                This is actually a boot loader parameter; the value is
                passed to the kernel using a special protocol.

    vmalloc=nn[KMG] [KNL,BOOT] Forces the vmalloc area to have an exact
                size of <nn>. This can be used to increase the
```

```
                minimum size (128MB on x86). It can also be used to
                decrease the size and leave more room for directly
                mapped kernel RAM.

vmcp_cma=nn[MG] [KNL,S390]
                Sets the memory size reserved for contiguous memory
                allocations for the vmcp device driver.

vmhalt=         [KNL,S390] Perform z/VM CP command after system halt.
                Format: <command>

vmpanic=        [KNL,S390] Perform z/VM CP command after kernel panic.
                Format: <command>

vmpoff=         [KNL,S390] Perform z/VM CP command after power off.
                Format: <command>

vsyscall=       [X86-64]
                Controls the behavior of vsyscalls (i.e. calls to
                fixed addresses of 0xffffffffff600x00 from legacy
                code).  Most statically-linked binaries and older
                versions of glibc use these calls.  Because these
                functions are at fixed addresses, they make nice
                targets for exploits that can control RIP.

                emulate     [default] Vsyscalls turn into traps and are
                            emulated reasonably safely.

                native      Vsyscalls are native syscall instructions.
                            This is a little bit faster than trapping
                            and makes a few dynamic recompilers work
                            better than they would in emulation mode.
                            It also makes exploits much easier to write.

                none        Vsyscalls don't work at all.  This makes
                            them quite hard to use for exploits but
                            might break your system.

vt.color=       [VT] Default text color.
                Format: 0xYX, X = foreground, Y = background.
                Default: 0x07 = light gray on black.

vt.cur_default= [VT] Default cursor shape.
                Format: 0xCCBBAA, where AA, BB, and CC are the same as
                the parameters of the <Esc>[?A;B;Cc escape sequence;
                see VGA-softcursor.txt. Default: 2 = underline.

vt.default_blu= [VT]
                Format: <blue0>,<blue1>,<blue2>,...,<blue15>
                Change the default blue palette of the console.
                This is a 16-member array composed of values
                ranging from 0-255.

vt.default_grn= [VT]
                Format: <green0>,<green1>,<green2>,...,<green15>
                Change the default green palette of the console.
                This is a 16-member array composed of values
                ranging from 0-255.
```

```
vt.default_red= [VT]
                Format: <red0>,<red1>,<red2>,...,<red15>
                Change the default red palette of the console.
                This is a 16-member array composed of values
                ranging from 0-255.

vt.default_utf8=
                [VT]
                Format=<0|1>
                Set system-wide default UTF-8 mode for all tty's.
                Default is 1, i.e. UTF-8 mode is enabled for all
                newly opened terminals.

vt.global_cursor_default=
                [VT]
                Format=<-1|0|1>
                Set system-wide default for whether a cursor
                is shown on new VTs. Default is -1,
                i.e. cursors will be created by default unless
                overridden by individual drivers. 0 will hide
                cursors, 1 will display them.

vt.italic=      [VT] Default color for italic text; 0-15.
                Default: 2 = green.

vt.underline=   [VT] Default color for underlined text; 0-15.
                Default: 3 = cyan.

watchdog timers [HW,WDT] For information on watchdog timers,
                see Documentation/watchdog/watchdog-parameters.txt
                or other driver-specific files in the
                Documentation/watchdog/ directory.

workqueue.watchdog_thresh=
                If CONFIG_WQ_WATCHDOG is configured, workqueue can
                warn stall conditions and dump internal state to
                help debugging.  0 disables workqueue stall
                detection; otherwise, it's the stall threshold
                duration in seconds.  The default value is 30 and
                it can be updated at runtime by writing to the
                corresponding sysfs file.

workqueue.disable_numa
                By default, all work items queued to unbound
                workqueues are affine to the NUMA nodes they're
                issued on, which results in better behavior in
                general.  If NUMA affinity needs to be disabled for
                whatever reason, this option can be used.  Note
                that this also can be controlled per-workqueue for
                workqueues visible under /sys/bus/workqueue/.

workqueue.power_efficient
                Per-cpu workqueues are generally preferred because
                they show better performance thanks to cache
                locality; unfortunately, per-cpu workqueues tend to
                be more power hungry than unbound workqueues.
```

```
                Enabling this makes the per-cpu workqueues which
                were observed to contribute significantly to power
                consumption unbound, leading to measurably lower
                power usage at the cost of small performance
                overhead.

                The default value of this parameter is determined by
                the config option CONFIG_WQ_POWER_EFFICIENT_DEFAULT.

workqueue.debug_force_rr_cpu
                Workqueue used to implicitly guarantee that work
                items queued without explicit CPU specified are put
                on the local CPU.  This guarantee is no longer true
                and while local CPU is still preferred work items
                may be put on foreign CPUs.  This debug option
                forces round-robin CPU selection to flush out
                usages which depend on the now broken guarantee.
                When enabled, memory and cache locality will be
                impacted.

x2apic_phys     [X86-64,APIC] Use x2apic physical mode instead of
                default x2apic cluster mode on platforms
                supporting x2apic.

x86_intel_mid_timer= [X86-32,APBT]
                Choose timer option for x86 Intel MID platform.
                Two valid options are apbt timer only and lapic timer
                plus one apbt timer for broadcast timer.
                x86_intel_mid_timer=apbt_only | lapic_and_apbt

xen_512gb_limit         [KNL,X86-64,XEN]
                Restricts the kernel running paravirtualized under Xen
                to use only up to 512 GB of RAM. The reason to do so is
                crash analysis tools and Xen tools for doing domain
                save/restore/migration must be enabled to handle larger
                domains.

xen_emul_unplug=                [HW,X86,XEN]
                Unplug Xen emulated devices
                Format: [unplug0,][unplug1]
                ide-disks -- unplug primary master IDE devices
                aux-ide-disks -- unplug non-primary-master IDE devices
                nics -- unplug network devices
                all -- unplug all emulated devices (NICs and IDE disks)
                unnecessary -- unplugging emulated devices is
                        unnecessary even if the host did not respond to
                        the unplug protocol
                never -- do not unplug even if version check succeeds

xen_nopvspin    [X86,XEN]
                Disables the ticketlock slowpath using Xen PV
                optimizations.

xen_nopv        [X86]
                Disables the PV optimizations forcing the HVM guest to
                run as generic HVM guest with no PV drivers.

xirc2ps_cs=     [NET,PCMCIA]
```

```
                        Format:
                        <irq>,<irq_mask>,<io>,<full_duplex>,<do_sound>,<lockup_hack>[,<irq2>[
```

Todo

Add more DRM drivers.

CHAPTER

THREE

LINUX ALLOCATED DEVICES (4.X+ VERSION)

This list is the Linux Device List, the official registry of allocated device numbers and /dev directory nodes for the Linux operating system.

The LaTeX version of this document is no longer maintained, nor is the document that used to reside at lanana.org. This version in the mainline Linux kernel is the master document. Updates shall be sent as patches to the kernel maintainers (see the Documentation/process/submitting-patches.rst document). Specifically explore the sections titled "CHAR and MISC DRIVERS", and "BLOCK LAYER" in the MAINTAINERS file to find the right maintainers to involve for character and block devices.

This document is included by reference into the Filesystem Hierarchy Standard (FHS). The FHS is available from http://www.pathname.com/fhs/.

Allocations marked (68k/Amiga) apply to Linux/68k on the Amiga platform only. Allocations marked (68k/Atari) apply to Linux/68k on the Atari platform only.

Attention:

DEVICE DRIVERS AUTHORS PLEASE READ THIS
Linux now has extensive support for dynamic allocation of device numbering and can use sysfs and udev (systemd) to handle the naming needs. There are still some exceptions in the serial and boot device area. Before asking for a device number make sure you actually need one.
To have a major number allocated, or a minor number in situations where that applies (e.g. bus-mice), please submit a patch and send to the authors as indicated above.
Keep the description of the device in the same format as this list. The reason for this is that it is the only way we have found to ensure we have all the requisite information to publish your device and avoid conflicts.
Finally, sometimes we have to play "namespace police." Please don't be offended. We often get submissions for /dev names that would be bound to cause conflicts down the road. We are trying to avoid getting in a situation where we would have to suffer an incompatible forward change. Therefore, please consult with us ***before*** *you make your device names and numbers in any way public, at least to the point where it would be at all difficult to get them changed.*
Your cooperation is appreciated.

```
0             Unnamed devices (e.g. non-device mounts)
                0 = reserved as null device number
              See block major 144, 145, 146 for expansion areas.

1 char        Memory devices
                1 = /dev/mem          Physical memory access
                2 = /dev/kmem         Kernel virtual memory access
                3 = /dev/null         Null device
                4 = /dev/port         I/O port access
                5 = /dev/zero         Null byte source
```

```
              6 = /dev/core         OBSOLETE - replaced by /proc/kcore
              7 = /dev/full         Returns ENOSPC on write
              8 = /dev/random       Nondeterministic random number gen.
              9 = /dev/urandom      Faster, less secure random number gen.
             10 = /dev/aio          Asynchronous I/O notification interface
             11 = /dev/kmsg         Writes to this come out as printk's, reads
                                    export the buffered printk records.
             12 = /dev/oldmem       OBSOLETE - replaced by /proc/vmcore

1 block      RAM disk
              0 = /dev/ram0         First RAM disk
              1 = /dev/ram1         Second RAM disk
                ...
            250 = /dev/initrd       Initial RAM disk

            Older kernels had /dev/ramdisk (1, 1) here.
            /dev/initrd refers to a RAM disk which was preloaded
            by the boot loader; newer kernels use /dev/ram0 for
            the initrd.

2 char       Pseudo-TTY masters
              0 = /dev/ptyp0        First PTY master
              1 = /dev/ptyp1        Second PTY master
                ...
            255 = /dev/ptyef        256th PTY master

            Pseudo-tty's are named as follows:
            * Masters are ``pty'', slaves are ``tty'';
            * the fourth letter is one of pqrstuvwxyzabcde indicating
              the 1st through 16th series of 16 pseudo-ttys each, and
            * the fifth letter is one of 0123456789abcdef indicating
              the position within the series.

            These are the old-style (BSD) PTY devices; Unix98
            devices are on major 128 and above and use the PTY
            master multiplex (/dev/ptmx) to acquire a PTY on
            demand.

2 block      Floppy disks
              0 = /dev/fd0          Controller 0, drive 0, autodetect
              1 = /dev/fd1          Controller 0, drive 1, autodetect
              2 = /dev/fd2          Controller 0, drive 2, autodetect
              3 = /dev/fd3          Controller 0, drive 3, autodetect
            128 = /dev/fd4          Controller 1, drive 0, autodetect
            129 = /dev/fd5          Controller 1, drive 1, autodetect
            130 = /dev/fd6          Controller 1, drive 2, autodetect
            131 = /dev/fd7          Controller 1, drive 3, autodetect

            To specify format, add to the autodetect device number:
              0 = /dev/fd?          Autodetect format
              4 = /dev/fd?d360      5.25''  360K in a 360K  drive(1)
             20 = /dev/fd?h360      5.25''  360K in a 1200K drive(1)
             48 = /dev/fd?h410      5.25''  410K in a 1200K drive
             64 = /dev/fd?h420      5.25''  420K in a 1200K drive
             24 = /dev/fd?h720      5.25''  720K in a 1200K drive
             80 = /dev/fd?h880      5.25''  880K in a 1200K drive(1)
              8 = /dev/fd?h1200     5.25'' 1200K in a 1200K drive(1)
             40 = /dev/fd?h1440     5.25'' 1440K in a 1200K drive(1)
```

```
              56 = /dev/fd?h1476     5.25'' 1476K in a 1200K drive
              72 = /dev/fd?h1494     5.25'' 1494K in a 1200K drive
              92 = /dev/fd?h1600     5.25'' 1600K in a 1200K drive(1)

              12 = /dev/fd?u360      3.5''   360K Double Density(2)
              16 = /dev/fd?u720      3.5''   720K Double Density(1)
             120 = /dev/fd?u800      3.5''   800K Double Density(2)
              52 = /dev/fd?u820      3.5''   820K Double Density
              68 = /dev/fd?u830      3.5''   830K Double Density
              84 = /dev/fd?u1040     3.5''  1040K Double Density(1)
              88 = /dev/fd?u1120     3.5''  1120K Double Density(1)
              28 = /dev/fd?u1440     3.5''  1440K High Density(1)
             124 = /dev/fd?u1600     3.5''  1600K High Density(1)
              44 = /dev/fd?u1680     3.5''  1680K High Density(3)
              60 = /dev/fd?u1722     3.5''  1722K High Density
              76 = /dev/fd?u1743     3.5''  1743K High Density
              96 = /dev/fd?u1760     3.5''  1760K High Density
             116 = /dev/fd?u1840     3.5''  1840K High Density(3)
             100 = /dev/fd?u1920     3.5''  1920K High Density(1)
              32 = /dev/fd?u2880     3.5''  2880K Extra Density(1)
             104 = /dev/fd?u3200     3.5''  3200K Extra Density
             108 = /dev/fd?u3520     3.5''  3520K Extra Density
             112 = /dev/fd?u3840     3.5''  3840K Extra Density(1)

              36 = /dev/fd?CompaQ    Compaq 2880K drive; obsolete?

             (1) Autodetectable format
             (2) Autodetectable format in a Double Density (720K) drive only
             (3) Autodetectable format in a High Density (1440K) drive only

             NOTE: The letter in the device name (d, q, h or u)
             signifies the type of drive: 5.25'' Double Density (d),
             5.25'' Quad Density (q), 5.25'' High Density (h) or 3.5''
             (any model, u).  The use of the capital letters D, H
             and E for the 3.5'' models have been deprecated, since
             the drive type is insignificant for these devices.

3 char       Pseudo-TTY slaves
               0 = /dev/ttyp0        First PTY slave
               1 = /dev/ttyp1        Second PTY slave
                 ...
             255 = /dev/ttyef        256th PTY slave

             These are the old-style (BSD) PTY devices; Unix98
             devices are on major 136 and above.

3 block      First MFM, RLL and IDE hard disk/CD-ROM interface
               0 = /dev/hda          Master: whole disk (or CD-ROM)
              64 = /dev/hdb          Slave: whole disk (or CD-ROM)

             For partitions, add to the whole disk device number:
               0 = /dev/hd?          Whole disk
               1 = /dev/hd?1         First partition
               2 = /dev/hd?2         Second partition
                 ...
              63 = /dev/hd?63        63rd partition

             For Linux/i386, partitions 1-4 are the primary
```

```
              partitions, and 5 and above are logical partitions.
              Other versions of Linux use partitioning schemes
              appropriate to their respective architectures.

  4 char      TTY devices
                0 = /dev/tty0         Current virtual console

                1 = /dev/tty1         First virtual console
                  ...
               63 = /dev/tty63        63rd virtual console
               64 = /dev/ttyS0        First UART serial port
                  ...
              255 = /dev/ttyS191      192nd UART serial port

              UART serial ports refer to 8250/16450/16550 series devices.

              Older versions of the Linux kernel used this major
              number for BSD PTY devices.  As of Linux 2.1.115, this
              is no longer supported.  Use major numbers 2 and 3.

  4 block     Aliases for dynamically allocated major devices to be used
              when its not possible to create the real device nodes
              because the root filesystem is mounted read-only.

                 0 = /dev/root

  5 char      Alternate TTY devices
                0 = /dev/tty          Current TTY device
                1 = /dev/console      System console
                2 = /dev/ptmx         PTY master multiplex
                3 = /dev/ttyprintk    User messages via printk TTY device
               64 = /dev/cua0         Callout device for ttyS0
                  ...
              255 = /dev/cua191       Callout device for ttyS191

              (5,1) is /dev/console starting with Linux 2.1.71.  See
              the section on terminal devices for more information
              on /dev/console.

  6 char      Parallel printer devices
                0 = /dev/lp0          Parallel printer on parport0
                1 = /dev/lp1          Parallel printer on parport1
                  ...

              Current Linux kernels no longer have a fixed mapping
              between parallel ports and I/O addresses.  Instead,
              they are redirected through the parport multiplex layer.

  7 char      Virtual console capture devices
                0 = /dev/vcs          Current vc text contents
                1 = /dev/vcs1         tty1 text contents
                  ...
               63 = /dev/vcs63        tty63 text contents
              128 = /dev/vcsa         Current vc text/attribute contents
              129 = /dev/vcsa1        tty1 text/attribute contents
                  ...
              191 = /dev/vcsa63       tty63 text/attribute contents
```

```
              NOTE: These devices permit both read and write access.

7 block       Loopback devices
                0 = /dev/loop0        First loop device
                1 = /dev/loop1        Second loop device
                  ...

              The loop devices are used to mount filesystems not
              associated with block devices.  The binding to the
              loop devices is handled by mount(8) or losetup(8).

8 block       SCSI disk devices (0-15)
                0 = /dev/sda          First SCSI disk whole disk
               16 = /dev/sdb          Second SCSI disk whole disk
               32 = /dev/sdc          Third SCSI disk whole disk
                  ...
              240 = /dev/sdp          Sixteenth SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

9 char        SCSI tape devices
                0 = /dev/st0          First SCSI tape, mode 0
                1 = /dev/st1          Second SCSI tape, mode 0
                  ...
               32 = /dev/st0l         First SCSI tape, mode 1
               33 = /dev/st1l         Second SCSI tape, mode 1
                  ...
               64 = /dev/st0m         First SCSI tape, mode 2
               65 = /dev/st1m         Second SCSI tape, mode 2
                  ...
               96 = /dev/st0a         First SCSI tape, mode 3
               97 = /dev/st1a         Second SCSI tape, mode 3
                    ...
              128 = /dev/nst0         First SCSI tape, mode 0, no rewind
              129 = /dev/nst1         Second SCSI tape, mode 0, no rewind
                  ...
              160 = /dev/nst0l        First SCSI tape, mode 1, no rewind
              161 = /dev/nst1l        Second SCSI tape, mode 1, no rewind
                  ...
              192 = /dev/nst0m        First SCSI tape, mode 2, no rewind
              193 = /dev/nst1m        Second SCSI tape, mode 2, no rewind
                  ...
              224 = /dev/nst0a        First SCSI tape, mode 3, no rewind
              225 = /dev/nst1a        Second SCSI tape, mode 3, no rewind
                  ...

              ``No rewind'' refers to the omission of the default
              automatic rewind on device close.  The MTREW or MTOFFL
              ioctl()'s can be used to rewind the tape regardless of
              the device used to access it.

9 block       Metadisk (RAID) devices
                0 = /dev/md0          First metadisk group
                1 = /dev/md1          Second metadisk group
                  ...
```

```
                The metadisk driver is used to span a
                filesystem across multiple physical disks.

  10 char       Non-serial mice, misc features
                  0 = /dev/logibm       Logitech bus mouse
                  1 = /dev/psaux        PS/2-style mouse port
                  2 = /dev/inportbm     Microsoft Inport bus mouse
                  3 = /dev/atibm        ATI XL bus mouse
                  4 = /dev/jbm          J-mouse
                  4 = /dev/amigamouse   Amiga mouse (68k/Amiga)
                  5 = /dev/atarimouse   Atari mouse
                  6 = /dev/sunmouse     Sun mouse
                  7 = /dev/amigamouse1  Second Amiga mouse
                  8 = /dev/smouse       Simple serial mouse driver
                  9 = /dev/pc110pad     IBM PC-110 digitizer pad
                 10 = /dev/adbmouse     Apple Desktop Bus mouse
                 11 = /dev/vrtpanel     Vr41xx embedded touch panel
                 13 = /dev/vpcmouse     Connectix Virtual PC Mouse
                 14 = /dev/touchscreen/ucb1x00  UCB 1x00 touchscreen
                 15 = /dev/touchscreen/mk712    MK712 touchscreen
                128 = /dev/beep         Fancy beep device
                129 =
                130 = /dev/watchdog     Watchdog timer port
                131 = /dev/temperature  Machine internal temperature
                132 = /dev/hwtrap       Hardware fault trap
                133 = /dev/exttrp       External device trap
                134 = /dev/apm_bios     Advanced Power Management BIOS
                135 = /dev/rtc          Real Time Clock
                137 = /dev/vhci         Bluetooth virtual HCI driver
                139 = /dev/openprom     SPARC OpenBoot PROM
                140 = /dev/relay8       Berkshire Products Octal relay card
                141 = /dev/relay16      Berkshire Products ISO-16 relay card
                142 =
                143 = /dev/pciconf      PCI configuration space
                144 = /dev/nvram        Non-volatile configuration RAM
                145 = /dev/hfmodem      Soundcard shortwave modem control
                146 = /dev/graphics     Linux/SGI graphics device
                147 = /dev/opengl       Linux/SGI OpenGL pipe
                148 = /dev/gfx          Linux/SGI graphics effects device
                149 = /dev/input/mouse  Linux/SGI Irix emulation mouse
                150 = /dev/input/keyboard Linux/SGI Irix emulation keyboard
                151 = /dev/led          Front panel LEDs
                152 = /dev/kpoll        Kernel Poll Driver
                153 = /dev/mergemem     Memory merge device
                154 = /dev/pmu          Macintosh PowerBook power manager
                155 = /dev/isictl       MultiTech ISICom serial control
                156 = /dev/lcd          Front panel LCD display
                157 = /dev/ac           Applicom Intl Profibus card
                158 = /dev/nwbutton     Netwinder external button
                159 = /dev/nwdebug      Netwinder debug interface
                160 = /dev/nwflash      Netwinder flash memory
                161 = /dev/userdma      User-space DMA access
                162 = /dev/smbus        System Management Bus
                163 = /dev/lik          Logitech Internet Keyboard
                164 = /dev/ipmo         Intel Intelligent Platform Management
                165 = /dev/vmmon        VMware virtual machine monitor
                166 = /dev/i2o/ctl      I2O configuration manager
                167 = /dev/specialix_sxctl Specialix serial control
```

```
168 = /dev/tcldrv       Technology Concepts serial control
169 = /dev/specialix_rioctl Specialix RIO serial control
170 = /dev/thinkpad/thinkpad    IBM Thinkpad devices
171 = /dev/srripc       QNX4 API IPC manager
172 = /dev/usemaclone   Semaphore clone device
173 = /dev/ipmikcs      Intelligent Platform Management
174 = /dev/uctrl        SPARCbook 3 microcontroller
175 = /dev/agpgart      AGP Graphics Address Remapping Table
176 = /dev/gtrsc        Gorgy Timing radio clock
177 = /dev/cbm          Serial CBM bus
178 = /dev/jsflash      JavaStation OS flash SIMM
179 = /dev/xsvc         High-speed shared-mem/semaphore service
180 = /dev/vrbuttons    Vr41xx button input device
181 = /dev/toshiba      Toshiba laptop SMM support
182 = /dev/perfctr      Performance-monitoring counters
183 = /dev/hwrng        Generic random number generator
184 = /dev/cpu/microcode CPU microcode update interface
186 = /dev/atomicps     Atomic shapshot of process state data
187 = /dev/irnet        IrNET device
188 = /dev/smbusbios    SMBus BIOS
189 = /dev/ussp_ctl     User space serial port control
190 = /dev/crash        Mission Critical Linux crash dump facility
191 = /dev/pcl181       <information missing>
192 = /dev/nas_xbus     NAS xbus LCD/buttons access
193 = /dev/d7s          SPARC 7-segment display
194 = /dev/zkshim       Zero-Knowledge network shim control
195 = /dev/elographics/e2201    Elographics touchscreen E271-2201
196 = /dev/vfio/vfio    VFIO userspace driver interface
197 = /dev/pxa3xx-gcu   PXA3xx graphics controller unit driver
198 = /dev/sexec        Signed executable interface
199 = /dev/scanners/cuecat :CueCat barcode scanner
200 = /dev/net/tun      TAP/TUN network device
201 = /dev/button/gulpb Transmeta GULP-B buttons
202 = /dev/emd/ctl      Enhanced Metadisk RAID (EMD) control
203 = /dev/cuse         Cuse (character device in user-space)
204 = /dev/video/em8300         EM8300 DVD decoder control
205 = /dev/video/em8300_mv      EM8300 DVD decoder video
206 = /dev/video/em8300_ma      EM8300 DVD decoder audio
207 = /dev/video/em8300_sp      EM8300 DVD decoder subpicture
208 = /dev/compaq/cpqphpc       Compaq PCI Hot Plug Controller
209 = /dev/compaq/cpqrid        Compaq Remote Insight Driver
210 = /dev/impi/bt      IMPI coprocessor block transfer
211 = /dev/impi/smic    IMPI coprocessor stream interface
212 = /dev/watchdogs/0  First watchdog device
213 = /dev/watchdogs/1  Second watchdog device
214 = /dev/watchdogs/2  Third watchdog device
215 = /dev/watchdogs/3  Fourth watchdog device
216 = /dev/fujitsu/apanel       Fujitsu/Siemens application panel
217 = /dev/ni/natmotn           National Instruments Motion
218 = /dev/kchuid       Inter-process chuid control
219 = /dev/modems/mwave MWave modem firmware upload
220 = /dev/mptctl       Message passing technology (MPT) control
221 = /dev/mvista/hssdsi        Montavista PICMG hot swap system driver
222 = /dev/mvista/hasi          Montavista PICMG high availability
223 = /dev/input/uinput         User level driver support for input
224 = /dev/tpm          TCPA TPM driver
225 = /dev/pps          Pulse Per Second driver
226 = /dev/systrace     Systrace device
```

```
                227 = /dev/mcelog       X86_64 Machine Check Exception driver
                228 = /dev/hpet         HPET driver
                229 = /dev/fuse         Fuse (virtual filesystem in user-space)
                230 = /dev/midishare    MidiShare driver
                231 = /dev/snapshot     System memory snapshot device
                232 = /dev/kvm          Kernel-based virtual machine (hardware virtual-
ization extensions)
                233 = /dev/kmview       View-OS A process with a view
                234 = /dev/btrfs-control        Btrfs control device
                235 = /dev/autofs       Autofs control device
                236 = /dev/mapper/control       Device-Mapper control device
                237 = /dev/loop-control Loopback control device
                238 = /dev/vhost-net    Host kernel accelerator for virtio net
                239 = /dev/uhid         User-space I/O driver support for HID subsystem
                240 = /dev/userio       Serio driver testing device
                241 = /dev/vhost-vsock  Host kernel driver for virtio vsock

                242-254                 Reserved for local use
                255                     Reserved for MISC_DYNAMIC_MINOR

  11 char       Raw keyboard device     (Linux/SPARC only)
                  0 = /dev/kbd          Raw keyboard device

  11 char       Serial Mux device       (Linux/PA-RISC only)
                  0 = /dev/ttyB0        First mux port
                  1 = /dev/ttyB1        Second mux port
                    ...

  11 block      SCSI CD-ROM devices
                  0 = /dev/scd0         First SCSI CD-ROM
                  1 = /dev/scd1         Second SCSI CD-ROM
                    ...

                The prefix /dev/sr (instead of /dev/scd) has been deprecated.

  12 char       QIC-02 tape
                  2 = /dev/ntpqic11     QIC-11, no rewind-on-close
                  3 = /dev/tpqic11      QIC-11, rewind-on-close
                  4 = /dev/ntpqic24     QIC-24, no rewind-on-close
                  5 = /dev/tpqic24      QIC-24, rewind-on-close
                  6 = /dev/ntpqic120    QIC-120, no rewind-on-close
                  7 = /dev/tpqic120     QIC-120, rewind-on-close
                  8 = /dev/ntpqic150    QIC-150, no rewind-on-close
                  9 = /dev/tpqic150     QIC-150, rewind-on-close

                The device names specified are proposed -- if there
                are ``standard'' names for these devices, please let me know.

  12 block

  13 char       Input core
                  0 = /dev/input/js0    First joystick
                  1 = /dev/input/js1    Second joystick
                    ...
                 32 = /dev/input/mouse0 First mouse
                 33 = /dev/input/mouse1 Second mouse
                    ...
                 63 = /dev/input/mice   Unified mouse
```

```
                64 = /dev/input/event0 First event queue
                65 = /dev/input/event1 Second event queue
                   ...

               Each device type has 5 bits (32 minors).

13 block       Previously used for the XT disk (/dev/xdN)
               Deleted in kernel v3.9.

14 char        Open Sound System (OSS)
                 0 = /dev/mixer        Mixer control
                 1 = /dev/sequencer    Audio sequencer
                 2 = /dev/midi00       First MIDI port
                 3 = /dev/dsp          Digital audio
                 4 = /dev/audio        Sun-compatible digital audio
                 6 =
                 7 = /dev/audioctl     SPARC audio control device
                 8 = /dev/sequencer2   Sequencer -- alternate device
                16 = /dev/mixer1       Second soundcard mixer control
                17 = /dev/patmgr0      Sequencer patch manager
                18 = /dev/midi01       Second MIDI port
                19 = /dev/dsp1         Second soundcard digital audio
                20 = /dev/audio1       Second soundcard Sun digital audio
                33 = /dev/patmgr1      Sequencer patch manager
                34 = /dev/midi02       Third MIDI port
                50 = /dev/midi03       Fourth MIDI port

14 block

15 char        Joystick
                 0 = /dev/js0          First analog joystick
                 1 = /dev/js1          Second analog joystick
                   ...
               128 = /dev/djs0         First digital joystick
               129 = /dev/djs1         Second digital joystick
                   ...
15 block       Sony CDU-31A/CDU-33A CD-ROM
                 0 = /dev/sonycd       Sony CDU-31a CD-ROM

16 char        Non-SCSI scanners
                 0 = /dev/gs4500       Genius 4500 handheld scanner

16 block       GoldStar CD-ROM
                 0 = /dev/gscd         GoldStar CD-ROM

17 char        OBSOLETE (was Chase serial card)
                 0 = /dev/ttyH0        First Chase port
                 1 = /dev/ttyH1        Second Chase port
                   ...
17 block       Optics Storage CD-ROM
                 0 = /dev/optcd        Optics Storage CD-ROM

18 char        OBSOLETE (was Chase serial card - alternate devices)
                 0 = /dev/cuh0         Callout device for ttyH0
                 1 = /dev/cuh1         Callout device for ttyH1
                   ...
18 block       Sanyo CD-ROM
                 0 = /dev/sjcd         Sanyo CD-ROM
```

```
19 char       Cyclades serial card
                0 = /dev/ttyC0        First Cyclades port
                  ...
               31 = /dev/ttyC31       32nd Cyclades port

19 block      ``Double'' compressed disk
                0 = /dev/double0      First compressed disk
                  ...
                7 = /dev/double7      Eighth compressed disk
              128 = /dev/cdouble0     Mirror of first compressed disk
                  ...
              135 = /dev/cdouble7     Mirror of eighth compressed disk

              See the Double documentation for the meaning of the
              mirror devices.

20 char       Cyclades serial card - alternate devices
                0 = /dev/cub0         Callout device for ttyC0
                  ...
               31 = /dev/cub31        Callout device for ttyC31

20 block      Hitachi CD-ROM (under development)
                0 = /dev/hitcd        Hitachi CD-ROM

21 char       Generic SCSI access
                0 = /dev/sg0          First generic SCSI device
                1 = /dev/sg1          Second generic SCSI device
                  ...

              Most distributions name these /dev/sga, /dev/sgb...;
              this sets an unnecessary limit of 26 SCSI devices in
              the system and is counter to standard Linux
              device-naming practice.

21 block      Acorn MFM hard drive interface
                0 = /dev/mfma         First MFM drive whole disk
               64 = /dev/mfmb         Second MFM drive whole disk

              This device is used on the ARM-based Acorn RiscPC.
              Partitions are handled the same way as for IDE disks
              (see major number 3).

22 char       Digiboard serial card
                0 = /dev/ttyD0        First Digiboard port
                1 = /dev/ttyD1        Second Digiboard port
                  ...
22 block      Second IDE hard disk/CD-ROM interface
                0 = /dev/hdc          Master: whole disk (or CD-ROM)
               64 = /dev/hdd          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

23 char       Digiboard serial card - alternate devices
                0 = /dev/cud0         Callout device for ttyD0
                1 = /dev/cud1         Callout device for ttyD1
                    ...
```

```
23 block      Mitsumi proprietary CD-ROM
                0 = /dev/mcd          Mitsumi CD-ROM

24 char       Stallion serial card
                0 = /dev/ttyE0        Stallion port 0 card 0
                1 = /dev/ttyE1        Stallion port 1 card 0
                  ...
               64 = /dev/ttyE64       Stallion port 0 card 1
               65 = /dev/ttyE65       Stallion port 1 card 1
                  ...
              128 = /dev/ttyE128      Stallion port 0 card 2
              129 = /dev/ttyE129      Stallion port 1 card 2
                  ...
              192 = /dev/ttyE192      Stallion port 0 card 3
              193 = /dev/ttyE193      Stallion port 1 card 3
                  ...
24 block      Sony CDU-535 CD-ROM
                0 = /dev/cdu535       Sony CDU-535 CD-ROM

25 char       Stallion serial card - alternate devices
                0 = /dev/cue0         Callout device for ttyE0
                1 = /dev/cue1         Callout device for ttyE1
                  ...
               64 = /dev/cue64        Callout device for ttyE64
               65 = /dev/cue65        Callout device for ttyE65
                  ...
              128 = /dev/cue128       Callout device for ttyE128
              129 = /dev/cue129       Callout device for ttyE129
                  ...
              192 = /dev/cue192       Callout device for ttyE192
              193 = /dev/cue193       Callout device for ttyE193
                  ...
25 block      First Matsushita (Panasonic/SoundBlaster) CD-ROM
                0 = /dev/sbpcd0       Panasonic CD-ROM controller 0 unit 0
                1 = /dev/sbpcd1       Panasonic CD-ROM controller 0 unit 1
                2 = /dev/sbpcd2       Panasonic CD-ROM controller 0 unit 2
                3 = /dev/sbpcd3       Panasonic CD-ROM controller 0 unit 3

26 char

26 block      Second Matsushita (Panasonic/SoundBlaster) CD-ROM
                0 = /dev/sbpcd4       Panasonic CD-ROM controller 1 unit 0
                1 = /dev/sbpcd5       Panasonic CD-ROM controller 1 unit 1
                2 = /dev/sbpcd6       Panasonic CD-ROM controller 1 unit 2
                3 = /dev/sbpcd7       Panasonic CD-ROM controller 1 unit 3

27 char       QIC-117 tape
                0 = /dev/qft0         Unit 0, rewind-on-close
                1 = /dev/qft1         Unit 1, rewind-on-close
                2 = /dev/qft2         Unit 2, rewind-on-close
                3 = /dev/qft3         Unit 3, rewind-on-close
                4 = /dev/nqft0        Unit 0, no rewind-on-close
                5 = /dev/nqft1        Unit 1, no rewind-on-close
                6 = /dev/nqft2        Unit 2, no rewind-on-close
                7 = /dev/nqft3        Unit 3, no rewind-on-close
               16 = /dev/zqft0        Unit 0, rewind-on-close, compression
               17 = /dev/zqft1        Unit 1, rewind-on-close, compression
               18 = /dev/zqft2        Unit 2, rewind-on-close, compression
```

```
                19 = /dev/zqft3        Unit 3, rewind-on-close, compression
                20 = /dev/nzqft0       Unit 0, no rewind-on-close, compression
                21 = /dev/nzqft1       Unit 1, no rewind-on-close, compression
                22 = /dev/nzqft2       Unit 2, no rewind-on-close, compression
                23 = /dev/nzqft3       Unit 3, no rewind-on-close, compression
                32 = /dev/rawqft0      Unit 0, rewind-on-close, no file marks
                33 = /dev/rawqft1      Unit 1, rewind-on-close, no file marks
                34 = /dev/rawqft2      Unit 2, rewind-on-close, no file marks
                35 = /dev/rawqft3      Unit 3, rewind-on-close, no file marks
                36 = /dev/nrawqft0     Unit 0, no rewind-on-close, no file marks
                37 = /dev/nrawqft1     Unit 1, no rewind-on-close, no file marks
                38 = /dev/nrawqft2     Unit 2, no rewind-on-close, no file marks
                39 = /dev/nrawqft3     Unit 3, no rewind-on-close, no file marks

  27 block      Third Matsushita (Panasonic/SoundBlaster) CD-ROM
                  0 = /dev/sbpcd8      Panasonic CD-ROM controller 2 unit 0
                  1 = /dev/sbpcd9      Panasonic CD-ROM controller 2 unit 1
                  2 = /dev/sbpcd10     Panasonic CD-ROM controller 2 unit 2
                  3 = /dev/sbpcd11     Panasonic CD-ROM controller 2 unit 3

  28 char       Stallion serial card - card programming
                  0 = /dev/staliomem0  First Stallion card I/O memory
                  1 = /dev/staliomem1  Second Stallion card I/O memory
                  2 = /dev/staliomem2  Third Stallion card I/O memory
                  3 = /dev/staliomem3  Fourth Stallion card I/O memory

  28 char       Atari SLM ACSI laser printer (68k/Atari)
                  0 = /dev/slm0        First SLM laser printer
                  1 = /dev/slm1        Second SLM laser printer
                    ...
  28 block      Fourth Matsushita (Panasonic/SoundBlaster) CD-ROM
                  0 = /dev/sbpcd12     Panasonic CD-ROM controller 3 unit 0
                  1 = /dev/sbpcd13     Panasonic CD-ROM controller 3 unit 1
                  2 = /dev/sbpcd14     Panasonic CD-ROM controller 3 unit 2
                  3 = /dev/sbpcd15     Panasonic CD-ROM controller 3 unit 3

  28 block      ACSI disk (68k/Atari)
                  0 = /dev/ada         First ACSI disk whole disk
                 16 = /dev/adb         Second ACSI disk whole disk
                 32 = /dev/adc         Third ACSI disk whole disk
                    ...
                240 = /dev/adp         16th ACSI disk whole disk

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15, like SCSI.

  29 char       Universal frame buffer
                  0 = /dev/fb0         First frame buffer
                  1 = /dev/fb1         Second frame buffer
                    ...
                 31 = /dev/fb31        32nd frame buffer

  29 block      Aztech/Orchid/Okano/Wearnes CD-ROM
                  0 = /dev/aztcd       Aztech CD-ROM

  30 char       iBCS-2 compatibility devices
                  0 = /dev/socksys     Socket access
```

```
                1 = /dev/spx          SVR3 local X interface
               32 = /dev/inet/ip      Network access
               33 = /dev/inet/icmp
               34 = /dev/inet/ggp
               35 = /dev/inet/ipip
               36 = /dev/inet/tcp
               37 = /dev/inet/egp
               38 = /dev/inet/pup
               39 = /dev/inet/udp
               40 = /dev/inet/idp
               41 = /dev/inet/rawip

             Additionally, iBCS-2 requires the following links:

             /dev/ip -> /dev/inet/ip
             /dev/icmp -> /dev/inet/icmp
             /dev/ggp -> /dev/inet/ggp
             /dev/ipip -> /dev/inet/ipip
             /dev/tcp -> /dev/inet/tcp
             /dev/egp -> /dev/inet/egp
             /dev/pup -> /dev/inet/pup
             /dev/udp -> /dev/inet/udp
             /dev/idp -> /dev/inet/idp
             /dev/rawip -> /dev/inet/rawip
             /dev/inet/arp -> /dev/inet/udp
             /dev/inet/rip -> /dev/inet/udp
             /dev/nfsd -> /dev/socksys
             /dev/X0R -> /dev/null (? apparently not required ?)

30 block     Philips LMS CM-205 CD-ROM
               0 = /dev/cm205cd      Philips LMS CM-205 CD-ROM

             /dev/lmscd is an older name for this device.  This
             driver does not work with the CM-205MS CD-ROM.

31 char      MPU-401 MIDI
               0 = /dev/mpu401data   MPU-401 data port
               1 = /dev/mpu401stat   MPU-401 status port

31 block     ROM/flash memory card
               0 = /dev/rom0         First ROM card (rw)
                  ...
               7 = /dev/rom7         Eighth ROM card (rw)
               8 = /dev/rrom0        First ROM card (ro)
                  ...
              15 = /dev/rrom7        Eighth ROM card (ro)
              16 = /dev/flash0       First flash memory card (rw)
                  ...
              23 = /dev/flash7       Eighth flash memory card (rw)
              24 = /dev/rflash0      First flash memory card (ro)
                  ...
              31 = /dev/rflash7      Eighth flash memory card (ro)

             The read-write (rw) devices support back-caching
             written data in RAM, as well as writing to flash RAM
             devices.  The read-only devices (ro) support reading
             only.
```

```
32 char       Specialix serial card
                0 = /dev/ttyX0        First Specialix port
                1 = /dev/ttyX1        Second Specialix port
                  ...
32 block      Philips LMS CM-206 CD-ROM
                0 = /dev/cm206cd      Philips LMS CM-206 CD-ROM

33 char       Specialix serial card - alternate devices
                0 = /dev/cux0         Callout device for ttyX0
                1 = /dev/cux1         Callout device for ttyX1
                  ...
33 block      Third IDE hard disk/CD-ROM interface
                0 = /dev/hde          Master: whole disk (or CD-ROM)
               64 = /dev/hdf          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

34 char       Z8530 HDLC driver
                0 = /dev/scc0         First Z8530, first port
                1 = /dev/scc1         First Z8530, second port
                2 = /dev/scc2         Second Z8530, first port
                3 = /dev/scc3         Second Z8530, second port
                  ...

              In a previous version these devices were named
              /dev/sc1 for /dev/scc0, /dev/sc2 for /dev/scc1, and so
              on.

34 block      Fourth IDE hard disk/CD-ROM interface
                0 = /dev/hdg          Master: whole disk (or CD-ROM)
               64 = /dev/hdh          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

35 char       tclmidi MIDI driver
                0 = /dev/midi0        First MIDI port, kernel timed
                1 = /dev/midi1        Second MIDI port, kernel timed
                2 = /dev/midi2        Third MIDI port, kernel timed
                3 = /dev/midi3        Fourth MIDI port, kernel timed
               64 = /dev/rmidi0       First MIDI port, untimed
               65 = /dev/rmidi1       Second MIDI port, untimed
               66 = /dev/rmidi2       Third MIDI port, untimed
               67 = /dev/rmidi3       Fourth MIDI port, untimed
              128 = /dev/smpte0       First MIDI port, SMPTE timed
              129 = /dev/smpte1       Second MIDI port, SMPTE timed
              130 = /dev/smpte2       Third MIDI port, SMPTE timed
              131 = /dev/smpte3       Fourth MIDI port, SMPTE timed

35 block      Slow memory ramdisk
                0 = /dev/slram        Slow memory ramdisk

36 char       Netlink support
                0 = /dev/route        Routing, device updates, kernel to user
                1 = /dev/skip         enSKIP security cache control
                3 = /dev/fwmonitor    Firewall packet copies
               16 = /dev/tap0         First Ethertap device
```

```
              ...
               31 = /dev/tap15        16th Ethertap device

36 block      OBSOLETE (was MCA ESDI hard disk)

37 char       IDE tape
                0 = /dev/ht0          First IDE tape
                1 = /dev/ht1          Second IDE tape
                ...
              128 = /dev/nht0         First IDE tape, no rewind-on-close
              129 = /dev/nht1         Second IDE tape, no rewind-on-close
                ...

              Currently, only one IDE tape drive is supported.

37 block      Zorro II ramdisk
                0 = /dev/z2ram        Zorro II ramdisk

38 char       Myricom PCI Myrinet board
                0 = /dev/mlanai0      First Myrinet board
                1 = /dev/mlanai1      Second Myrinet board
                ...

              This device is used for status query, board control
              and ``user level packet I/O.''  This board is also
              accessible as a standard networking ``eth'' device.

38 block      OBSOLETE (was Linux/AP+)

39 char       ML-16P experimental I/O board
                0 = /dev/ml16pa-a0    First card, first analog channel
                1 = /dev/ml16pa-a1    First card, second analog channel
                ...
               15 = /dev/ml16pa-a15   First card, 16th analog channel
               16 = /dev/ml16pa-d     First card, digital lines
               17 = /dev/ml16pa-c0    First card, first counter/timer
               18 = /dev/ml16pa-c1    First card, second counter/timer
               19 = /dev/ml16pa-c2    First card, third counter/timer
               32 = /dev/ml16pb-a0    Second card, first analog channel
               33 = /dev/ml16pb-a1    Second card, second analog channel
                ...
               47 = /dev/ml16pb-a15   Second card, 16th analog channel
               48 = /dev/ml16pb-d     Second card, digital lines
               49 = /dev/ml16pb-c0    Second card, first counter/timer
               50 = /dev/ml16pb-c1    Second card, second counter/timer
               51 = /dev/ml16pb-c2    Second card, third counter/timer
                  ...
39 block

40 char

40 block

41 char       Yet Another Micro Monitor
                0 = /dev/yamm         Yet Another Micro Monitor

41 block
```

```
42 char       Demo/sample use

42 block      Demo/sample use

              This number is intended for use in sample code, as
              well as a general ``example'' device number.  It
              should never be used for a device driver that is being
              distributed; either obtain an official number or use
              the local/experimental range.  The sudden addition or
              removal of a driver with this number should not cause
              ill effects to the system (bugs excepted.)

              IN PARTICULAR, ANY DISTRIBUTION WHICH CONTAINS A
              DEVICE DRIVER USING MAJOR NUMBER 42 IS NONCOMPLIANT.

43 char       isdn4linux virtual modem
                0 = /dev/ttyI0        First virtual modem
                ...
               63 = /dev/ttyI63       64th virtual modem

43 block      Network block devices
                0 = /dev/nb0          First network block device
                1 = /dev/nb1          Second network block device
                ...

              Network Block Device is somehow similar to loopback
              devices: If you read from it, it sends packet across
              network asking server for data. If you write to it, it
              sends packet telling server to write. It could be used
              to mounting filesystems over the net, swapping over
              the net, implementing block device in userland etc.

44 char       isdn4linux virtual modem - alternate devices
                0 = /dev/cui0         Callout device for ttyI0
                ...
               63 = /dev/cui63        Callout device for ttyI63

44 block      Flash Translation Layer (FTL) filesystems
                0 = /dev/ftla         FTL on first Memory Technology Device
               16 = /dev/ftlb         FTL on second Memory Technology Device
               32 = /dev/ftlc         FTL on third Memory Technology Device
                ...
              240 = /dev/ftlp         FTL on 16th Memory Technology Device

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the partition
              limit is 15 rather than 63 per disk (same as SCSI.)

45 char       isdn4linux ISDN BRI driver
                0 = /dev/isdn0        First virtual B channel raw data
                ...
               63 = /dev/isdn63       64th virtual B channel raw data
               64 = /dev/isdnctrl0    First channel control/debug
                ...
              127 = /dev/isdnctrl63   64th channel control/debug

              128 = /dev/ippp0        First SyncPPP device
                ...
```

```
              191 = /dev/ippp63       64th SyncPPP device

              255 = /dev/isdninfo     ISDN monitor interface

45 block      Parallel port IDE disk devices
                0 = /dev/pda          First parallel port IDE disk
               16 = /dev/pdb          Second parallel port IDE disk
               32 = /dev/pdc          Third parallel port IDE disk
               48 = /dev/pdd          Fourth parallel port IDE disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the partition
              limit is 15 rather than 63 per disk.

46 char       Comtrol Rocketport serial card
                0 = /dev/ttyR0        First Rocketport port
                1 = /dev/ttyR1        Second Rocketport port
                  ...
46 block      Parallel port ATAPI CD-ROM devices
                0 = /dev/pcd0         First parallel port ATAPI CD-ROM
                1 = /dev/pcd1         Second parallel port ATAPI CD-ROM
                2 = /dev/pcd2         Third parallel port ATAPI CD-ROM
                3 = /dev/pcd3         Fourth parallel port ATAPI CD-ROM

47 char       Comtrol Rocketport serial card - alternate devices
                0 = /dev/cur0         Callout device for ttyR0
                1 = /dev/cur1         Callout device for ttyR1
                  ...
47 block      Parallel port ATAPI disk devices
                0 = /dev/pf0          First parallel port ATAPI disk
                1 = /dev/pf1          Second parallel port ATAPI disk
                2 = /dev/pf2          Third parallel port ATAPI disk
                3 = /dev/pf3          Fourth parallel port ATAPI disk

              This driver is intended for floppy disks and similar
              devices and hence does not support partitioning.

48 char       SDL RISCom serial card
                0 = /dev/ttyL0        First RISCom port
                1 = /dev/ttyL1        Second RISCom port
                  ...
48 block      Mylex DAC960 PCI RAID controller; first controller
                0 = /dev/rd/c0d0      First disk, whole disk
                8 = /dev/rd/c0d1      Second disk, whole disk
                  ...
              248 = /dev/rd/c0d31     32nd disk, whole disk

              For partitions add:
                0 = /dev/rd/c?d?      Whole disk
                1 = /dev/rd/c?d?p1    First partition
                  ...
                7 = /dev/rd/c?d?p7    Seventh partition

49 char       SDL RISCom serial card - alternate devices
                0 = /dev/cul0         Callout device for ttyL0
                1 = /dev/cul1         Callout device for ttyL1
                  ...
49 block      Mylex DAC960 PCI RAID controller; second controller
```

```
                0 = /dev/rd/c1d0      First disk, whole disk
                8 = /dev/rd/c1d1      Second disk, whole disk
                  ...
              248 = /dev/rd/c1d31     32nd disk, whole disk

              Partitions are handled as for major 48.

50 char       Reserved for GLINT

50 block      Mylex DAC960 PCI RAID controller; third controller
                0 = /dev/rd/c2d0      First disk, whole disk
                8 = /dev/rd/c2d1      Second disk, whole disk
                  ...
              248 = /dev/rd/c2d31     32nd disk, whole disk

51 char       Baycom radio modem OR Radio Tech BIM-XXX-RS232 radio modem
                0 = /dev/bc0          First Baycom radio modem
                1 = /dev/bc1          Second Baycom radio modem
                  ...
51 block      Mylex DAC960 PCI RAID controller; fourth controller
                0 = /dev/rd/c3d0      First disk, whole disk
                8 = /dev/rd/c3d1      Second disk, whole disk
                  ...
              248 = /dev/rd/c3d31     32nd disk, whole disk

              Partitions are handled as for major 48.

52 char       Spellcaster DataComm/BRI ISDN card
                0 = /dev/dcbri0       First DataComm card
                1 = /dev/dcbri1       Second DataComm card
                2 = /dev/dcbri2       Third DataComm card
                3 = /dev/dcbri3       Fourth DataComm card

52 block      Mylex DAC960 PCI RAID controller; fifth controller
                0 = /dev/rd/c4d0      First disk, whole disk
                8 = /dev/rd/c4d1      Second disk, whole disk
                  ...
              248 = /dev/rd/c4d31     32nd disk, whole disk

              Partitions are handled as for major 48.

53 char       BDM interface for remote debugging MC683xx microcontrollers
                0 = /dev/pd_bdm0      PD BDM interface on lp0
                1 = /dev/pd_bdm1      PD BDM interface on lp1
                2 = /dev/pd_bdm2      PD BDM interface on lp2
                4 = /dev/icd_bdm0     ICD BDM interface on lp0
                5 = /dev/icd_bdm1     ICD BDM interface on lp1
                6 = /dev/icd_bdm2     ICD BDM interface on lp2

              This device is used for the interfacing to the MC683xx
              microcontrollers via Background Debug Mode by use of a
              Parallel Port interface. PD is the Motorola Public
              Domain Interface and ICD is the commercial interface
              by P&E.

53 block      Mylex DAC960 PCI RAID controller; sixth controller
                0 = /dev/rd/c5d0      First disk, whole disk
                8 = /dev/rd/c5d1      Second disk, whole disk
```

```
                ...
              248 = /dev/rd/c5d31     32nd disk, whole disk

              Partitions are handled as for major 48.

54 char       Electrocardiognosis Holter serial card
                0 = /dev/holter0      First Holter port
                1 = /dev/holter1      Second Holter port
                2 = /dev/holter2      Third Holter port

              A custom serial card used by Electrocardiognosis SRL
              <mseritan@ottonel.pub.ro> to transfer data from Holter
              24-hour heart monitoring equipment.

54 block      Mylex DAC960 PCI RAID controller; seventh controller
                0 = /dev/rd/c6d0      First disk, whole disk
                8 = /dev/rd/c6d1      Second disk, whole disk
                ...
              248 = /dev/rd/c6d31     32nd disk, whole disk

              Partitions are handled as for major 48.

55 char       DSP56001 digital signal processor
                0 = /dev/dsp56k       First DSP56001

55 block      Mylex DAC960 PCI RAID controller; eighth controller
                0 = /dev/rd/c7d0      First disk, whole disk
                8 = /dev/rd/c7d1      Second disk, whole disk
                ...
              248 = /dev/rd/c7d31     32nd disk, whole disk

              Partitions are handled as for major 48.

56 char       Apple Desktop Bus
                0 = /dev/adb          ADB bus control

              Additional devices will be added to this number, all
              starting with /dev/adb.

56 block      Fifth IDE hard disk/CD-ROM interface
                0 = /dev/hdi          Master: whole disk (or CD-ROM)
               64 = /dev/hdj          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

57 char       Hayes ESP serial card
                0 = /dev/ttyP0        First ESP port
                1 = /dev/ttyP1        Second ESP port
                  ...

57 block      Sixth IDE hard disk/CD-ROM interface
                0 = /dev/hdk          Master: whole disk (or CD-ROM)
               64 = /dev/hdl          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).
```

```
58 char       Hayes ESP serial card - alternate devices
                0 = /dev/cup0         Callout device for ttyP0
                1 = /dev/cup1         Callout device for ttyP1
                  ...

58 block      Reserved for logical volume manager

59 char       sf firewall package
                0 = /dev/firewall     Communication with sf kernel module

59 block      Generic PDA filesystem device
                0 = /dev/pda0         First PDA device
                1 = /dev/pda1         Second PDA device
                  ...

              The pda devices are used to mount filesystems on
              remote pda's (basically slow handheld machines with
              proprietary OS's and limited memory and storage
              running small fs translation drivers) through serial /
              IRDA / parallel links.

              NAMING CONFLICT -- PROPOSED REVISED NAME /dev/rpda0 etc

60-63 char    LOCAL/EXPERIMENTAL USE

60-63 block   LOCAL/EXPERIMENTAL USE
              Allocated for local/experimental use.  For devices not
              assigned official numbers, these ranges should be
              used in order to avoid conflicting with future assignments.

64 char       ENskip kernel encryption package
                0 = /dev/enskip       Communication with ENskip kernel module

64 block      Scramdisk/DriveCrypt encrypted devices
                0 = /dev/scramdisk/master    Master node for ioctls
                1 = /dev/scramdisk/1         First encrypted device
                2 = /dev/scramdisk/2         Second encrypted device
                ...
              255 = /dev/scramdisk/255       255th encrypted device

              The filename of the encrypted container and the passwords
              are sent via ioctls (using the sdmount tool) to the master
              node which then activates them via one of the
              /dev/scramdisk/x nodes for loop mounting (all handled
              through the sdmount tool).

              Requested by: andy@scramdisklinux.org

65 char       Sundance ``plink'' Transputer boards (obsolete, unused)
                0 = /dev/plink0       First plink device
                1 = /dev/plink1       Second plink device
                2 = /dev/plink2       Third plink device
                3 = /dev/plink3       Fourth plink device
               64 = /dev/rplink0      First plink device, raw
               65 = /dev/rplink1      Second plink device, raw
               66 = /dev/rplink2      Third plink device, raw
               67 = /dev/rplink3      Fourth plink device, raw
              128 = /dev/plink0d      First plink device, debug
```

```
              129 = /dev/plink1d      Second plink device, debug
              130 = /dev/plink2d      Third plink device, debug
              131 = /dev/plink3d      Fourth plink device, debug
              192 = /dev/rplink0d     First plink device, raw, debug
              193 = /dev/rplink1d     Second plink device, raw, debug
              194 = /dev/rplink2d     Third plink device, raw, debug
              195 = /dev/rplink3d     Fourth plink device, raw, debug

              This is a commercial driver; contact James Howes
              <jth@prosig.demon.co.uk> for information.

65 block      SCSI disk devices (16-31)
                0 = /dev/sdq          17th SCSI disk whole disk
               16 = /dev/sdr          18th SCSI disk whole disk
               32 = /dev/sds          19th SCSI disk whole disk
                  ...
              240 = /dev/sdaf         32nd SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

66 char       YARC PowerPC PCI coprocessor card
                0 = /dev/yppcpci0     First YARC card
                1 = /dev/yppcpci1     Second YARC card
                  ...

66 block      SCSI disk devices (32-47)
                0 = /dev/sdag         33th SCSI disk whole disk
               16 = /dev/sdah         34th SCSI disk whole disk
               32 = /dev/sdai         35th SCSI disk whole disk
                  ...
              240 = /dev/sdav         48nd SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

67 char       Coda network file system
                0 = /dev/cfs0         Coda cache manager

              See http://www.coda.cs.cmu.edu for information about Coda.

67 block      SCSI disk devices (48-63)
                0 = /dev/sdaw         49th SCSI disk whole disk
               16 = /dev/sdax         50th SCSI disk whole disk
               32 = /dev/sday         51st SCSI disk whole disk
                  ...
              240 = /dev/sdbl         64th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

68 char       CAPI 2.0 interface
                0 = /dev/capi20       Control device
                1 = /dev/capi20.00    First CAPI 2.0 application
                2 = /dev/capi20.01    Second CAPI 2.0 application
```

```
                ...
               20 = /dev/capi20.19    19th CAPI 2.0 application

              ISDN CAPI 2.0 driver for use with CAPI 2.0
              applications; currently supports the AVM B1 card.

68 block      SCSI disk devices (64-79)
                0 = /dev/sdbm         65th SCSI disk whole disk
               16 = /dev/sdbn         66th SCSI disk whole disk
               32 = /dev/sdbo         67th SCSI disk whole disk
                ...
              240 = /dev/sdcb         80th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

69 char       MA16 numeric accelerator card
                0 = /dev/ma16         Board memory access

69 block      SCSI disk devices (80-95)
                0 = /dev/sdcc         81st SCSI disk whole disk
               16 = /dev/sdcd         82nd SCSI disk whole disk
               32 = /dev/sdce         83th SCSI disk whole disk
                ...
              240 = /dev/sdcr         96th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

70 char       SpellCaster Protocol Services Interface
                0 = /dev/apscfg       Configuration interface
                1 = /dev/apsauth      Authentication interface
                2 = /dev/apslog       Logging interface
                3 = /dev/apsdbg       Debugging interface
               64 = /dev/apsisdn      ISDN command interface
               65 = /dev/apsasync     Async command interface
              128 = /dev/apsmon       Monitor interface

70 block      SCSI disk devices (96-111)
                0 = /dev/sdcs         97th SCSI disk whole disk
               16 = /dev/sdct         98th SCSI disk whole disk
               32 = /dev/sdcu         99th SCSI disk whole disk
                ...
              240 = /dev/sddh         112nd SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

71 char       Computone IntelliPort II serial card
                0 = /dev/ttyF0        IntelliPort II board 0, port 0
                1 = /dev/ttyF1        IntelliPort II board 0, port 1
                ...
               63 = /dev/ttyF63       IntelliPort II board 0, port 63
               64 = /dev/ttyF64       IntelliPort II board 1, port 0
               65 = /dev/ttyF65       IntelliPort II board 1, port 1
```

```
                ...
               127 = /dev/ttyF127      IntelliPort II board 1, port 63
               128 = /dev/ttyF128      IntelliPort II board 2, port 0
               129 = /dev/ttyF129      IntelliPort II board 2, port 1
                ...
               191 = /dev/ttyF191      IntelliPort II board 2, port 63
               192 = /dev/ttyF192      IntelliPort II board 3, port 0
               193 = /dev/ttyF193      IntelliPort II board 3, port 1
                ...
               255 = /dev/ttyF255      IntelliPort II board 3, port 63

71 block       SCSI disk devices (112-127)
                 0 = /dev/sddi         113th SCSI disk whole disk
                16 = /dev/sddj         114th SCSI disk whole disk
                32 = /dev/sddk         115th SCSI disk whole disk
                ...
               240 = /dev/sddx         128th SCSI disk whole disk

               Partitions are handled in the same way as for IDE
               disks (see major number 3) except that the limit on
               partitions is 15.

72 char        Computone IntelliPort II serial card - alternate devices
                 0 = /dev/cuf0         Callout device for ttyF0
                 1 = /dev/cuf1         Callout device for ttyF1
                ...
                63 = /dev/cuf63        Callout device for ttyF63
                64 = /dev/cuf64        Callout device for ttyF64
                65 = /dev/cuf65        Callout device for ttyF65
                ...
               127 = /dev/cuf127       Callout device for ttyF127
               128 = /dev/cuf128       Callout device for ttyF128
               129 = /dev/cuf129       Callout device for ttyF129
                ...
               191 = /dev/cuf191       Callout device for ttyF191
               192 = /dev/cuf192       Callout device for ttyF192
               193 = /dev/cuf193       Callout device for ttyF193
                ...
               255 = /dev/cuf255       Callout device for ttyF255

72 block       Compaq Intelligent Drive Array, first controller
                 0 = /dev/ida/c0d0     First logical drive whole disk
                16 = /dev/ida/c0d1     Second logical drive whole disk
                ...
               240 = /dev/ida/c0d15    16th logical drive whole disk

               Partitions are handled the same way as for Mylex
               DAC960 (see major number 48) except that the limit on
               partitions is 15.

73 char        Computone IntelliPort II serial card - control devices
                 0 = /dev/ip2ipl0      Loadware device for board 0
                 1 = /dev/ip2stat0     Status device for board 0
                 4 = /dev/ip2ipl1      Loadware device for board 1
                 5 = /dev/ip2stat1     Status device for board 1
                 8 = /dev/ip2ipl2      Loadware device for board 2
                 9 = /dev/ip2stat2     Status device for board 2
                12 = /dev/ip2ipl3      Loadware device for board 3
```

```
               13 = /dev/ip2stat3     Status device for board 3

73 block       Compaq Intelligent Drive Array, second controller
                 0 = /dev/ida/c1d0     First logical drive whole disk
                16 = /dev/ida/c1d1     Second logical drive whole disk
                   ...
               240 = /dev/ida/c1d15    16th logical drive whole disk

               Partitions are handled the same way as for Mylex
               DAC960 (see major number 48) except that the limit on
               partitions is 15.

74 char        SCI bridge
                 0 = /dev/SCI/0        SCI device 0
                 1 = /dev/SCI/1        SCI device 1
                   ...

               Currently for Dolphin Interconnect Solutions' PCI-SCI
               bridge.

74 block       Compaq Intelligent Drive Array, third controller
                 0 = /dev/ida/c2d0     First logical drive whole disk
                16 = /dev/ida/c2d1     Second logical drive whole disk
                   ...
               240 = /dev/ida/c2d15    16th logical drive whole disk

               Partitions are handled the same way as for Mylex
               DAC960 (see major number 48) except that the limit on
               partitions is 15.

75 char        Specialix IO8+ serial card
                 0 = /dev/ttyW0        First IO8+ port, first card
                 1 = /dev/ttyW1        Second IO8+ port, first card
                   ...
                 8 = /dev/ttyW8        First IO8+ port, second card
                   ...

75 block       Compaq Intelligent Drive Array, fourth controller
                 0 = /dev/ida/c3d0     First logical drive whole disk
                16 = /dev/ida/c3d1     Second logical drive whole disk
                   ...
               240 = /dev/ida/c3d15    16th logical drive whole disk

               Partitions are handled the same way as for Mylex
               DAC960 (see major number 48) except that the limit on
               partitions is 15.

76 char        Specialix IO8+ serial card - alternate devices
                 0 = /dev/cuw0         Callout device for ttyW0
                 1 = /dev/cuw1         Callout device for ttyW1
                   ...
                 8 = /dev/cuw8         Callout device for ttyW8
                   ...

76 block       Compaq Intelligent Drive Array, fifth controller
                 0 = /dev/ida/c4d0     First logical drive whole disk
                16 = /dev/ida/c4d1     Second logical drive whole disk
                   ...
```

```
              240 = /dev/ida/c4d15    16th logical drive whole disk

              Partitions are handled the same way as for Mylex
              DAC960 (see major number 48) except that the limit on
              partitions is 15.


77 char       ComScire Quantum Noise Generator
                0 = /dev/qng          ComScire Quantum Noise Generator

77 block      Compaq Intelligent Drive Array, sixth controller
                0 = /dev/ida/c5d0     First logical drive whole disk
               16 = /dev/ida/c5d1     Second logical drive whole disk
                  ...
              240 = /dev/ida/c5d15    16th logical drive whole disk

              Partitions are handled the same way as for Mylex
              DAC960 (see major number 48) except that the limit on
              partitions is 15.

78 char       PAM Software's multimodem boards
                0 = /dev/ttyM0        First PAM modem
                1 = /dev/ttyM1        Second PAM modem
                  ...

78 block      Compaq Intelligent Drive Array, seventh controller
                0 = /dev/ida/c6d0     First logical drive whole disk
               16 = /dev/ida/c6d1     Second logical drive whole disk
                  ...
              240 = /dev/ida/c6d15    16th logical drive whole disk

              Partitions are handled the same way as for Mylex
              DAC960 (see major number 48) except that the limit on
              partitions is 15.

79 char       PAM Software's multimodem boards - alternate devices
                0 = /dev/cum0         Callout device for ttyM0
                1 = /dev/cum1         Callout device for ttyM1
                  ...

79 block      Compaq Intelligent Drive Array, eighth controller
                0 = /dev/ida/c7d0     First logical drive whole disk
               16 = /dev/ida/c7d1     Second logical drive whole disk
                  ...
              240 = /dev/ida/c715     16th logical drive whole disk

              Partitions are handled the same way as for Mylex
              DAC960 (see major number 48) except that the limit on
              partitions is 15.

80 char       Photometrics AT200 CCD camera
                0 = /dev/at200        Photometrics AT200 CCD camera

80 block      I2O hard disk
                0 = /dev/i2o/hda      First I2O hard disk, whole disk
               16 = /dev/i2o/hdb      Second I2O hard disk, whole disk
                  ...
              240 = /dev/i2o/hdp      16th I2O hard disk, whole disk
```

```
                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15.

  81 char       video4linux
                  0 = /dev/video0       Video capture/overlay device
                    ...
                 63 = /dev/video63      Video capture/overlay device
                 64 = /dev/radio0       Radio device
                    ...
                127 = /dev/radio63      Radio device
                128 = /dev/swradio0     Software Defined Radio device
                    ...
                191 = /dev/swradio63    Software Defined Radio device
                224 = /dev/vbi0         Vertical blank interrupt
                    ...
                255 = /dev/vbi31        Vertical blank interrupt

                Minor numbers are allocated dynamically unless
                CONFIG_VIDEO_FIXED_MINOR_RANGES (default n)
                configuration option is set.

  81 block      I2O hard disk
                  0 = /dev/i2o/hdq      17th I2O hard disk, whole disk
                 16 = /dev/i2o/hdr      18th I2O hard disk, whole disk
                    ...
                240 = /dev/i2o/hdaf     32nd I2O hard disk, whole disk

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15.

  82 char       WiNRADiO communications receiver card
                  0 = /dev/winradio0    First WiNRADiO card
                  1 = /dev/winradio1    Second WiNRADiO card
                    ...

                The driver and documentation may be obtained from
                http://www.winradio.com/

  82 block      I2O hard disk
                  0 = /dev/i2o/hdag     33rd I2O hard disk, whole disk
                 16 = /dev/i2o/hdah     34th I2O hard disk, whole disk
                    ...
                240 = /dev/i2o/hdav     48th I2O hard disk, whole disk

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15.

  83 char       Matrox mga_vid video driver
                 0 = /dev/mga_vid0      1st video card
                 1 = /dev/mga_vid1      2nd video card
                 2 = /dev/mga_vid2      3rd video card
                  ...
                15 = /dev/mga_vid15     16th video card
```

```
83 block      I2O hard disk
                0 = /dev/i2o/hdaw     49th I2O hard disk, whole disk
               16 = /dev/i2o/hdax     50th I2O hard disk, whole disk
                  ...
              240 = /dev/i2o/hdbl     64th I2O hard disk, whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

84 char       Ikon 1011[57] Versatec Greensheet Interface
                0 = /dev/ihcp0        First Greensheet port
                1 = /dev/ihcp1        Second Greensheet port

84 block      I2O hard disk
                0 = /dev/i2o/hdbm     65th I2O hard disk, whole disk
               16 = /dev/i2o/hdbn     66th I2O hard disk, whole disk
                  ...
              240 = /dev/i2o/hdcb     80th I2O hard disk, whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

85 char       Linux/SGI shared memory input queue
                0 = /dev/shmiq        Master shared input queue
                1 = /dev/qcntl0       First device pushed
                2 = /dev/qcntl1       Second device pushed
                  ...

85 block      I2O hard disk
                0 = /dev/i2o/hdcc     81st I2O hard disk, whole disk
               16 = /dev/i2o/hdcd     82nd I2O hard disk, whole disk
                  ...
              240 = /dev/i2o/hdcr     96th I2O hard disk, whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

86 char       SCSI media changer
                0 = /dev/sch0         First SCSI media changer
                1 = /dev/sch1         Second SCSI media changer
                  ...

86 block      I2O hard disk
                0 = /dev/i2o/hdcs     97th I2O hard disk, whole disk
               16 = /dev/i2o/hdct     98th I2O hard disk, whole disk
                  ...
              240 = /dev/i2o/hddh     112th I2O hard disk, whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

87 char       Sony Control-A1 stereo control bus
                0 = /dev/controla0    First device on chain
                1 = /dev/controla1    Second device on chain
```

```
                ...

87 block      I2O hard disk
                0 = /dev/i2o/hddi     113rd I2O hard disk, whole disk
               16 = /dev/i2o/hddj     114th I2O hard disk, whole disk
                ...
              240 = /dev/i2o/hddx     128th I2O hard disk, whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

88 char       COMX synchronous serial card
                0 = /dev/comx0        COMX channel 0
                1 = /dev/comx1        COMX channel 1
                ...

88 block      Seventh IDE hard disk/CD-ROM interface
                0 = /dev/hdm          Master: whole disk (or CD-ROM)
               64 = /dev/hdn          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

89 char       I2C bus interface
                0 = /dev/i2c-0        First I2C adapter
                1 = /dev/i2c-1        Second I2C adapter
                ...

89 block      Eighth IDE hard disk/CD-ROM interface
                0 = /dev/hdo          Master: whole disk (or CD-ROM)
               64 = /dev/hdp          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

90 char       Memory Technology Device (RAM, ROM, Flash)
                0 = /dev/mtd0         First MTD (rw)
                1 = /dev/mtdr0        First MTD (ro)
                ...
               30 = /dev/mtd15        16th MTD (rw)
               31 = /dev/mtdr15       16th MTD (ro)

90 block      Ninth IDE hard disk/CD-ROM interface
                0 = /dev/hdq          Master: whole disk (or CD-ROM)
               64 = /dev/hdr          Slave: whole disk (or CD-ROM)

              Partitions are handled the same way as for the first
              interface (see major number 3).

91 char       CAN-Bus devices
                0 = /dev/can0         First CAN-Bus controller
                1 = /dev/can1         Second CAN-Bus controller
                ...

91 block      Tenth IDE hard disk/CD-ROM interface
                0 = /dev/hds          Master: whole disk (or CD-ROM)
               64 = /dev/hdt          Slave: whole disk (or CD-ROM)
```

```
              Partitions are handled the same way as for the first
              interface (see major number 3).

92 char       Reserved for ith Kommunikationstechnik MIC ISDN card

92 block      PPDD encrypted disk driver
                0 = /dev/ppdd0        First encrypted disk
                1 = /dev/ppdd1        Second encrypted disk
                  ...

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

93 char

93 block      NAND Flash Translation Layer filesystem
                0 = /dev/nftla        First NFTL layer
               16 = /dev/nftlb        Second NFTL layer
                  ...
              240 = /dev/nftlp        16th NTFL layer

94 char

94 block      IBM S/390 DASD block storage
                0 = /dev/dasda First DASD device, major
                1 = /dev/dasda1 First DASD device, block 1
                2 = /dev/dasda2 First DASD device, block 2
                3 = /dev/dasda3 First DASD device, block 3
                4 = /dev/dasdb Second DASD device, major
                5 = /dev/dasdb1 Second DASD device, block 1
                6 = /dev/dasdb2 Second DASD device, block 2
                7 = /dev/dasdb3 Second DASD device, block 3
                  ...

95 char       IP filter
                0 = /dev/ipl          Filter control device/log file
                1 = /dev/ipnat        NAT control device/log file
                2 = /dev/ipstate      State information log file
                3 = /dev/ipauth       Authentication control device/log file
                  ...

96 char       Parallel port ATAPI tape devices
                0 = /dev/pt0          First parallel port ATAPI tape
                1 = /dev/pt1          Second parallel port ATAPI tape
                  ...
              128 = /dev/npt0         First p.p. ATAPI tape, no rewind
              129 = /dev/npt1         Second p.p. ATAPI tape, no rewind
                  ...

96 block      Inverse NAND Flash Translation Layer
                0 = /dev/inftla First INFTL layer
               16 = /dev/inftlb Second INFTL layer
                  ...
              240 = /dev/inftlp       16th INTFL layer

97 char       Parallel port generic ATAPI interface
```

```
                0 = /dev/pg0          First parallel port ATAPI device
                1 = /dev/pg1          Second parallel port ATAPI device
                2 = /dev/pg2          Third parallel port ATAPI device
                3 = /dev/pg3          Fourth parallel port ATAPI device

              These devices support the same API as the generic SCSI
              devices.

 98 char      Control and Measurement Device (comedi)
                0 = /dev/comedi0      First comedi device
                1 = /dev/comedi1      Second comedi device
                  ...

              See http://stm.lbl.gov/comedi.

 98 block     User-mode virtual block device
                0 = /dev/ubda         First user-mode block device
               16 = /dev/udbb         Second user-mode block device
                  ...

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

              This device is used by the user-mode virtual kernel port.

 99 char      Raw parallel ports
                0 = /dev/parport0     First parallel port
                1 = /dev/parport1     Second parallel port
                  ...

 99 block     JavaStation flash disk
                0 = /dev/jsfd         JavaStation flash disk

100 char      Telephony for Linux
                0 = /dev/phone0       First telephony device
                1 = /dev/phone1       Second telephony device
                  ...

101 char      Motorola DSP 56xxx board
                0 = /dev/mdspstat     Status information
                1 = /dev/mdsp1        First DSP board I/O controls
                  ...
               16 = /dev/mdsp16       16th DSP board I/O controls

101 block     AMI HyperDisk RAID controller
                0 = /dev/amiraid/ar0  First array whole disk
               16 = /dev/amiraid/ar1  Second array whole disk
                  ...
              240 = /dev/amiraid/ar15 16th array whole disk

              For each device, partitions are added as:
                0 = /dev/amiraid/ar?    Whole disk
                1 = /dev/amiraid/ar?p1  First partition
                2 = /dev/amiraid/ar?p2  Second partition
                  ...
               15 = /dev/amiraid/ar?p15 15th partition
```

```
102 char

102 block       Compressed block device
                  0 = /dev/cbd/a        First compressed block device, whole device
                 16 = /dev/cbd/b        Second compressed block device, whole device
                    ...
                240 = /dev/cbd/p        16th compressed block device, whole device

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15.

103 char        Arla network file system
                  0 = /dev/nnpfs0       First NNPFS device
                  1 = /dev/nnpfs1       Second NNPFS device

                Arla is a free clone of the Andrew File System, AFS.
                The NNPFS device gives user mode filesystem
                implementations a kernel presence for caching and easy
                mounting.  For more information about the project,
                write to <arla-drinkers@stacken.kth.se> or see
                http://www.stacken.kth.se/project/arla/

103 block       Audit device
                  0 = /dev/audit        Audit device

104 char        Flash BIOS support

104 block       Compaq Next Generation Drive Array, first controller
                  0 = /dev/cciss/c0d0   First logical drive, whole disk
                 16 = /dev/cciss/c0d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c0d15  16th logical drive, whole disk

                Partitions are handled the same way as for Mylex
                DAC960 (see major number 48) except that the limit on
                partitions is 15.

105 char        Comtrol VS-1000 serial controller
                  0 = /dev/ttyV0        First VS-1000 port
                  1 = /dev/ttyV1        Second VS-1000 port
                    ...

105 block       Compaq Next Generation Drive Array, second controller
                  0 = /dev/cciss/c1d0   First logical drive, whole disk
                 16 = /dev/cciss/c1d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c1d15  16th logical drive, whole disk

                Partitions are handled the same way as for Mylex
                DAC960 (see major number 48) except that the limit on
                partitions is 15.

106 char        Comtrol VS-1000 serial controller - alternate devices
                  0 = /dev/cuv0         First VS-1000 port
                  1 = /dev/cuv1         Second VS-1000 port
                    ...
```

```
106 block       Compaq Next Generation Drive Array, third controller
                  0 = /dev/cciss/c2d0   First logical drive, whole disk
                 16 = /dev/cciss/c2d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c2d15  16th logical drive, whole disk

                Partitions are handled the same way as for Mylex
                DAC960 (see major number 48) except that the limit on
                partitions is 15.

107 char        3Dfx Voodoo Graphics device
                  0 = /dev/3dfx         Primary 3Dfx graphics device

107 block       Compaq Next Generation Drive Array, fourth controller
                  0 = /dev/cciss/c3d0   First logical drive, whole disk
                 16 = /dev/cciss/c3d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c3d15  16th logical drive, whole disk

                Partitions are handled the same way as for Mylex
                DAC960 (see major number 48) except that the limit on
                partitions is 15.

108 char        Device independent PPP interface
                  0 = /dev/ppp          Device independent PPP interface

108 block       Compaq Next Generation Drive Array, fifth controller
                  0 = /dev/cciss/c4d0   First logical drive, whole disk
                 16 = /dev/cciss/c4d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c4d15  16th logical drive, whole disk

                Partitions are handled the same way as for Mylex
                DAC960 (see major number 48) except that the limit on
                partitions is 15.

109 char        Reserved for logical volume manager

109 block       Compaq Next Generation Drive Array, sixth controller
                  0 = /dev/cciss/c5d0   First logical drive, whole disk
                 16 = /dev/cciss/c5d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c5d15  16th logical drive, whole disk

                Partitions are handled the same way as for Mylex
                DAC960 (see major number 48) except that the limit on
                partitions is 15.

110 char        miroMEDIA Surround board
                  0 = /dev/srnd0        First miroMEDIA Surround board
                  1 = /dev/srnd1        Second miroMEDIA Surround board
                    ...

110 block       Compaq Next Generation Drive Array, seventh controller
                  0 = /dev/cciss/c6d0   First logical drive, whole disk
                 16 = /dev/cciss/c6d1   Second logical drive, whole disk
                    ...
                240 = /dev/cciss/c6d15  16th logical drive, whole disk
```

```
              Partitions are handled the same way as for Mylex
              DAC960 (see major number 48) except that the limit on
              partitions is 15.

111 char

111 block     Compaq Next Generation Drive Array, eighth controller
                0 = /dev/cciss/c7d0   First logical drive, whole disk
               16 = /dev/cciss/c7d1   Second logical drive, whole disk
                  ...
              240 = /dev/cciss/c7d15  16th logical drive, whole disk

              Partitions are handled the same way as for Mylex
              DAC960 (see major number 48) except that the limit on
              partitions is 15.

112 char      ISI serial card
                0 = /dev/ttyM0        First ISI port
                1 = /dev/ttyM1        Second ISI port
                  ...

              There is currently a device-naming conflict between
              these and PAM multimodems (major 78).

112 block     IBM iSeries virtual disk
                0 = /dev/iseries/vda  First virtual disk, whole disk
                8 = /dev/iseries/vdb  Second virtual disk, whole disk
                  ...
              200 = /dev/iseries/vdz  26th virtual disk, whole disk
              208 = /dev/iseries/vdaa 27th virtual disk, whole disk
                  ...
              248 = /dev/iseries/vdaf 32nd virtual disk, whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 7.

113 char      ISI serial card - alternate devices
                0 = /dev/cum0         Callout device for ttyM0
                1 = /dev/cum1         Callout device for ttyM1
                  ...

113 block     IBM iSeries virtual CD-ROM
                0 = /dev/iseries/vcda First virtual CD-ROM
                1 = /dev/iseries/vcdb Second virtual CD-ROM
                  ...

114 char      Picture Elements ISE board
                0 = /dev/ise0         First ISE board
                1 = /dev/ise1         Second ISE board
                  ...
              128 = /dev/isex0        Control node for first ISE board
              129 = /dev/isex1        Control node for second ISE board
                  ...

              The ISE board is an embedded computer, optimized for
              image processing. The /dev/iseN nodes are the general
```

```
              I/O access to the board, the /dev/isex0 nodes command
              nodes used to control the board.

114 block      IDE BIOS powered software RAID interfaces such as the
              Promise Fastrak

                 0 = /dev/ataraid/d0
                 1 = /dev/ataraid/d0p1
                 2 = /dev/ataraid/d0p2
                ...
                16 = /dev/ataraid/d1
                17 = /dev/ataraid/d1p1
                18 = /dev/ataraid/d1p2
                ...
               255 = /dev/ataraid/d15p15

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

115 char      TI link cable devices (115 was formerly the console driver speaker)
                0 = /dev/tipar0    Parallel cable on first parallel port
                ...
                7 = /dev/tipar7    Parallel cable on seventh parallel port

                8 = /dev/tiser0    Serial cable on first serial port
                ...
               15 = /dev/tiser7    Serial cable on seventh serial port

               16 = /dev/tiusb0    First USB cable
                ...
               47 = /dev/tiusb31   32nd USB cable

115 block      NetWare (NWFS) Devices (0-255)

              The NWFS (NetWare) devices are used to present a
              collection of NetWare Mirror Groups or NetWare
              Partitions as a logical storage segment for
              use in mounting NetWare volumes.  A maximum of
               256 NetWare volumes can be supported in a single
              machine.

              http://cgfa.telepac.pt/ftp2/kernel.org/linux/kernel/people/jmerkey/nwfs/

               0 = /dev/nwfs/v0    First NetWare (NWFS) Logical Volume
               1 = /dev/nwfs/v1    Second NetWare (NWFS) Logical Volume
               2 = /dev/nwfs/v2    Third NetWare (NWFS) Logical Volume
                    ...
               255 = /dev/nwfs/v255    Last NetWare (NWFS) Logical Volume

116 char      Advanced Linux Sound Driver (ALSA)

116 block      MicroMemory battery backed RAM adapter (NVRAM)
              Supports 16 boards, 15 partitions each.
              Requested by neilb at cse.unsw.edu.au.

               0 = /dev/umem/d0      Whole of first board
               1 = /dev/umem/d0p1    First partition of first board
```

```
                 2 = /dev/umem/d0p2    Second partition of first board
                15 = /dev/umem/d0p15   15th partition of first board

                16 = /dev/umem/d1      Whole of second board
                17 = /dev/umem/d1p1    First partition of second board
                   ...
               255= /dev/umem/d15p15  15th partition of 16th board.

117 char       COSA/SRP synchronous serial card
                 0 = /dev/cosa0c0      1st board, 1st channel
                 1 = /dev/cosa0c1      1st board, 2nd channel
                   ...
                16 = /dev/cosa1c0      2nd board, 1st channel
                17 = /dev/cosa1c1      2nd board, 2nd channel
                   ...

117 block       Enterprise Volume Management System (EVMS)

               The EVMS driver uses a layered, plug-in model to provide
               unparalleled flexibility and extensibility in managing
               storage.  This allows for easy expansion or customization
               of various levels of volume management.  Requested by
               Mark Peloquin (peloquin at us.ibm.com).

               Note: EVMS populates and manages all the devnodes in
               /dev/evms.

               http://sf.net/projects/evms

                  0 = /dev/evms/block_device   EVMS block device
                  1 = /dev/evms/legacyname1    First EVMS legacy device
                  2 = /dev/evms/legacyname2    Second EVMS legacy device
                   ...
                   Both ranges can grow (down or up) until they meet.
                   ...
                254 = /dev/evms/EVMSname2      Second EVMS native device
                255 = /dev/evms/EVMSname1      First EVMS native device

               Note: legacyname(s) are derived from the normal legacy
               device names.  For example, /dev/hda5 would become
               /dev/evms/hda5.

118 char       IBM Cryptographic Accelerator
                 0 = /dev/ica  Virtual interface to all IBM Crypto Accelerators
                 1 = /dev/ica0 IBMCA Device 0
                 2 = /dev/ica1 IBMCA Device 1
                   ...

119 char       VMware virtual network control
                 0 = /dev/vnet0        1st virtual network
                 1 = /dev/vnet1        2nd virtual network
                   ...

120-127 char   LOCAL/EXPERIMENTAL USE

120-127 block  LOCAL/EXPERIMENTAL USE
               Allocated for local/experimental use.  For devices not
               assigned official numbers, these ranges should be
```

```
              used in order to avoid conflicting with future assignments.

128-135 char  Unix98 PTY masters

              These devices should not have corresponding device
              nodes; instead they should be accessed through the
              /dev/ptmx cloning interface.

128 block      SCSI disk devices (128-143)
                 0 = /dev/sddy         129th SCSI disk whole disk
                16 = /dev/sddz         130th SCSI disk whole disk
                32 = /dev/sdea         131th SCSI disk whole disk
                   ...
               240 = /dev/sden         144th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

129 block      SCSI disk devices (144-159)
                 0 = /dev/sdeo         145th SCSI disk whole disk
                16 = /dev/sdep         146th SCSI disk whole disk
                32 = /dev/sdeq         147th SCSI disk whole disk
                   ...
               240 = /dev/sdfd         160th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

130 char      (Misc devices)

130 block      SCSI disk devices (160-175)
                 0 = /dev/sdfe         161st SCSI disk whole disk
                16 = /dev/sdff         162nd SCSI disk whole disk
                32 = /dev/sdfg         163rd SCSI disk whole disk
                   ...
               240 = /dev/sdft         176th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

131 block      SCSI disk devices (176-191)
                 0 = /dev/sdfu         177th SCSI disk whole disk
                16 = /dev/sdfv         178th SCSI disk whole disk
                32 = /dev/sdfw         179th SCSI disk whole disk
                   ...
               240 = /dev/sdgj         192nd SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

132 block      SCSI disk devices (192-207)
                 0 = /dev/sdgk         193rd SCSI disk whole disk
                16 = /dev/sdgl         194th SCSI disk whole disk
                32 = /dev/sdgm         195th SCSI disk whole disk
```

```
                ...
               240 = /dev/sdgz         208th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

133 block      SCSI disk devices (208-223)
                 0 = /dev/sdha         209th SCSI disk whole disk
                16 = /dev/sdhb         210th SCSI disk whole disk
                32 = /dev/sdhc         211th SCSI disk whole disk
                ...
               240 = /dev/sdhp         224th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

134 block      SCSI disk devices (224-239)
                 0 = /dev/sdhq         225th SCSI disk whole disk
                16 = /dev/sdhr         226th SCSI disk whole disk
                32 = /dev/sdhs         227th SCSI disk whole disk
                ...
               240 = /dev/sdif         240th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

135 block      SCSI disk devices (240-255)
                 0 = /dev/sdig         241st SCSI disk whole disk
                16 = /dev/sdih         242nd SCSI disk whole disk
                32 = /dev/sdih         243rd SCSI disk whole disk
                ...
               240 = /dev/sdiv         256th SCSI disk whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

136-143 char  Unix98 PTY slaves
                0 = /dev/pts/0        First Unix98 pseudo-TTY
                1 = /dev/pts/1        Second Unix98 pseudo-TTY
                  ...

              These device nodes are automatically generated with
              the proper permissions and modes by mounting the
              devpts filesystem onto /dev/pts with the appropriate
              mount options (distribution dependent, however, on
              *most* distributions the appropriate options are
              ``mode=0620,gid=<gid of the ``tty'' group>''.)

136 block     Mylex DAC960 PCI RAID controller; ninth controller
                0 = /dev/rd/c8d0      First disk, whole disk
                8 = /dev/rd/c8d1      Second disk, whole disk
                ...
              248 = /dev/rd/c8d31     32nd disk, whole disk
```

```
              Partitions are handled as for major 48.

137 block     Mylex DAC960 PCI RAID controller; tenth controller
                0 = /dev/rd/c9d0      First disk, whole disk
                8 = /dev/rd/c9d1      Second disk, whole disk
                  ...
              248 = /dev/rd/c9d31     32nd disk, whole disk

              Partitions are handled as for major 48.

138 block     Mylex DAC960 PCI RAID controller; eleventh controller
                0 = /dev/rd/c10d0     First disk, whole disk
                8 = /dev/rd/c10d1     Second disk, whole disk
                  ...
              248 = /dev/rd/c10d31    32nd disk, whole disk

              Partitions are handled as for major 48.

139 block     Mylex DAC960 PCI RAID controller; twelfth controller
                0 = /dev/rd/c11d0     First disk, whole disk
                8 = /dev/rd/c11d1     Second disk, whole disk
                  ...
              248 = /dev/rd/c11d31    32nd disk, whole disk

              Partitions are handled as for major 48.

140 block     Mylex DAC960 PCI RAID controller; thirteenth controller
                0 = /dev/rd/c12d0     First disk, whole disk
                8 = /dev/rd/c12d1     Second disk, whole disk
                  ...
              248 = /dev/rd/c12d31    32nd disk, whole disk

              Partitions are handled as for major 48.

141 block     Mylex DAC960 PCI RAID controller; fourteenth controller
                0 = /dev/rd/c13d0     First disk, whole disk
                8 = /dev/rd/c13d1     Second disk, whole disk
                  ...
              248 = /dev/rd/c13d31    32nd disk, whole disk

              Partitions are handled as for major 48.

142 block     Mylex DAC960 PCI RAID controller; fifteenth controller
                0 = /dev/rd/c14d0     First disk, whole disk
                8 = /dev/rd/c14d1     Second disk, whole disk
                  ...
              248 = /dev/rd/c14d31    32nd disk, whole disk

              Partitions are handled as for major 48.

143 block     Mylex DAC960 PCI RAID controller; sixteenth controller
                0 = /dev/rd/c15d0     First disk, whole disk
                8 = /dev/rd/c15d1     Second disk, whole disk
                  ...
              248 = /dev/rd/c15d31    32nd disk, whole disk

              Partitions are handled as for major 48.
```

```
144 char        Encapsulated PPP
                  0 = /dev/pppox0       First PPP over Ethernet
                    ...
                 63 = /dev/pppox63      64th PPP over Ethernet

                This is primarily used for ADSL.

                The SST 5136-DN DeviceNet interface driver has been
                relocated to major 183 due to an unfortunate conflict.

144 block       Expansion Area #1 for more non-device (e.g. NFS) mounts
                  0 = mounted device 256
                255 = mounted device 511

145 char        SAM9407-based soundcard
                  0 = /dev/sam0_mixer
                  1 = /dev/sam0_sequencer
                  2 = /dev/sam0_midi00
                  3 = /dev/sam0_dsp
                  4 = /dev/sam0_audio
                  6 = /dev/sam0_sndstat
                 18 = /dev/sam0_midi01
                 34 = /dev/sam0_midi02
                 50 = /dev/sam0_midi03
                 64 = /dev/sam1_mixer
                    ...
                128 = /dev/sam2_mixer
                    ...
                192 = /dev/sam3_mixer
                    ...

                Device functions match OSS, but offer a number of
                addons, which are sam9407 specific.  OSS can be
                operated simultaneously, taking care of the codec.

145 block       Expansion Area #2 for more non-device (e.g. NFS) mounts
                  0 = mounted device 512
                255 = mounted device 767

146 char        SYSTRAM SCRAMNet mirrored-memory network
                  0 = /dev/scramnet0    First SCRAMNet device
                  1 = /dev/scramnet1    Second SCRAMNet device
                    ...

146 block       Expansion Area #3 for more non-device (e.g. NFS) mounts
                  0 = mounted device 768
                255 = mounted device 1023

147 char        Aureal Semiconductor Vortex Audio device
                  0 = /dev/aureal0      First Aureal Vortex
                  1 = /dev/aureal1      Second Aureal Vortex
                    ...

147 block       Distributed Replicated Block Device (DRBD)
                  0 = /dev/drbd0        First DRBD device
                  1 = /dev/drbd1        Second DRBD device
                    ...
```

```
148 char        Technology Concepts serial card
                  0 = /dev/ttyT0        First TCL port
                  1 = /dev/ttyT1        Second TCL port
                    ...

149 char        Technology Concepts serial card - alternate devices
                  0 = /dev/cut0         Callout device for ttyT0
                  1 = /dev/cut0         Callout device for ttyT1
                    ...

150 char        Real-Time Linux FIFOs
                  0 = /dev/rtf0         First RTLinux FIFO
                  1 = /dev/rtf1         Second RTLinux FIFO
                    ...

151 char        DPT I2O SmartRaid V controller
                  0 = /dev/dpti0        First DPT I2O adapter
                  1 = /dev/dpti1        Second DPT I2O adapter
                    ...

152 char        EtherDrive Control Device
                  0 = /dev/etherd/ctl   Connect/Disconnect an EtherDrive
                  1 = /dev/etherd/err   Monitor errors
                  2 = /dev/etherd/raw   Raw AoE packet monitor

152 block       EtherDrive Block Devices
                  0 = /dev/etherd/0     EtherDrive 0
                    ...
                255 = /dev/etherd/255   EtherDrive 255

153 char        SPI Bus Interface (sometimes referred to as MicroWire)
                  0 = /dev/spi0         First SPI device on the bus
                  1 = /dev/spi1         Second SPI device on the bus
                    ...
                 15 = /dev/spi15        Sixteenth SPI device on the bus

153 block       Enhanced Metadisk RAID (EMD) storage units
                  0 = /dev/emd/0        First unit
                  1 = /dev/emd/0p1      Partition 1 on First unit
                  2 = /dev/emd/0p2      Partition 2 on First unit
                    ...
                 15 = /dev/emd/0p15     Partition 15 on First unit

                 16 = /dev/emd/1        Second unit
                 32 = /dev/emd/2        Third unit
                    ...
                240 = /dev/emd/15       Sixteenth unit

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 15.

154 char        Specialix RIO serial card
                  0 = /dev/ttySR0       First RIO port
                    ...
                255 = /dev/ttySR255     256th RIO port

155 char        Specialix RIO serial card - alternate devices
```

```
                0 = /dev/cusr0        Callout device for ttySR0
                ...
              255 = /dev/cusr255      Callout device for ttySR255

156 char      Specialix RIO serial card
                0 = /dev/ttySR256     257th RIO port
                ...
              255 = /dev/ttySR511     512th RIO port

157 char      Specialix RIO serial card - alternate devices
                0 = /dev/cusr256      Callout device for ttySR256
                ...
              255 = /dev/cusr511      Callout device for ttySR511

158 char      Dialogic GammaLink fax driver
                0 = /dev/gfax0        GammaLink channel 0
                1 = /dev/gfax1        GammaLink channel 1
                ...

159 char      RESERVED

159 block     RESERVED

160 char      General Purpose Instrument Bus (GPIB)
                0 = /dev/gpib0        First GPIB bus
                1 = /dev/gpib1        Second GPIB bus
                ...

160 block      Carmel 8-port SATA Disks on First Controller
                0 = /dev/carmel/0     SATA disk 0 whole disk
                1 = /dev/carmel/0p1   SATA disk 0 partition 1
                ...
               31 = /dev/carmel/0p31  SATA disk 0 partition 31

               32 = /dev/carmel/1     SATA disk 1 whole disk
               64 = /dev/carmel/2     SATA disk 2 whole disk
                ...
              224 = /dev/carmel/7     SATA disk 7 whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 31.

161 char      IrCOMM devices (IrDA serial/parallel emulation)
                0 = /dev/ircomm0      First IrCOMM device
                1 = /dev/ircomm1      Second IrCOMM device
                ...
               16 = /dev/irlpt0       First IrLPT device
               17 = /dev/irlpt1       Second IrLPT device
                ...

161 block      Carmel 8-port SATA Disks on Second Controller
                0 = /dev/carmel/8     SATA disk 8 whole disk
                1 = /dev/carmel/8p1   SATA disk 8 partition 1
                ...
               31 = /dev/carmel/8p31  SATA disk 8 partition 31

               32 = /dev/carmel/9     SATA disk 9 whole disk
```

```
                 64 = /dev/carmel/10    SATA disk 10 whole disk
                    ...
                224 = /dev/carmel/15    SATA disk 15 whole disk

                Partitions are handled in the same way as for IDE
                disks (see major number 3) except that the limit on
                partitions is 31.

162 char        Raw block device interface
                  0 = /dev/rawctl       Raw I/O control device
                  1 = /dev/raw/raw1     First raw I/O device
                  2 = /dev/raw/raw2     Second raw I/O device
                    ...
                 max minor number of raw device is set by kernel config
                 MAX_RAW_DEVS or raw module parameter `max_raw_devs'

163 char

164 char        Chase Research AT/PCI-Fast serial card
                  0 = /dev/ttyCH0       AT/PCI-Fast board 0, port 0
                    ...
                 15 = /dev/ttyCH15      AT/PCI-Fast board 0, port 15
                 16 = /dev/ttyCH16      AT/PCI-Fast board 1, port 0
                    ...
                 31 = /dev/ttyCH31      AT/PCI-Fast board 1, port 15
                 32 = /dev/ttyCH32      AT/PCI-Fast board 2, port 0
                    ...
                 47 = /dev/ttyCH47      AT/PCI-Fast board 2, port 15
                 48 = /dev/ttyCH48      AT/PCI-Fast board 3, port 0
                    ...
                 63 = /dev/ttyCH63      AT/PCI-Fast board 3, port 15

165 char        Chase Research AT/PCI-Fast serial card - alternate devices
                  0 = /dev/cuch0        Callout device for ttyCH0
                    ...
                 63 = /dev/cuch63       Callout device for ttyCH63

166 char        ACM USB modems
                  0 = /dev/ttyACM0      First ACM modem
                  1 = /dev/ttyACM1      Second ACM modem
                    ...

167 char        ACM USB modems - alternate devices
                  0 = /dev/cuacm0       Callout device for ttyACM0
                  1 = /dev/cuacm1       Callout device for ttyACM1
                    ...

168 char        Eracom CSA7000 PCI encryption adaptor
                  0 = /dev/ecsa0        First CSA7000
                  1 = /dev/ecsa1        Second CSA7000
                    ...

169 char        Eracom CSA8000 PCI encryption adaptor
                  0 = /dev/ecsa8-0      First CSA8000
                  1 = /dev/ecsa8-1      Second CSA8000
                    ...

170 char        AMI MegaRAC remote access controller
```

```
                0 = /dev/megarac0     First MegaRAC card
                1 = /dev/megarac1     Second MegaRAC card
                  ...

171 char        Reserved for IEEE 1394 (Firewire)

172 char        Moxa Intellio serial card
                0 = /dev/ttyMX0       First Moxa port
                1 = /dev/ttyMX1       Second Moxa port
                  ...
              127 = /dev/ttyMX127     128th Moxa port
              128 = /dev/moxactl      Moxa control port

173 char        Moxa Intellio serial card - alternate devices
                0 = /dev/cumx0        Callout device for ttyMX0
                1 = /dev/cumx1        Callout device for ttyMX1
                  ...
              127 = /dev/cumx127      Callout device for ttyMX127

174 char        SmartIO serial card
                0 = /dev/ttySI0       First SmartIO port
                1 = /dev/ttySI1       Second SmartIO port
                  ...

175 char        SmartIO serial card - alternate devices
                0 = /dev/cusi0        Callout device for ttySI0
                1 = /dev/cusi1        Callout device for ttySI1
                  ...

176 char        nCipher nFast PCI crypto accelerator
                0 = /dev/nfastpci0    First nFast PCI device
                1 = /dev/nfastpci1    First nFast PCI device
                  ...

177 char        TI PCILynx memory spaces
                0 = /dev/pcilynx/aux0  AUX space of first PCILynx card
                  ...
               15 = /dev/pcilynx/aux15 AUX space of 16th PCILynx card
               16 = /dev/pcilynx/rom0  ROM space of first PCILynx card
                  ...
               31 = /dev/pcilynx/rom15 ROM space of 16th PCILynx card
               32 = /dev/pcilynx/ram0  RAM space of first PCILynx card
                  ...
               47 = /dev/pcilynx/ram15 RAM space of 16th PCILynx card

178 char        Giganet cLAN1xxx virtual interface adapter
                0 = /dev/clanvi0      First cLAN adapter
                1 = /dev/clanvi1      Second cLAN adapter
                  ...

179 block        MMC block devices
                0 = /dev/mmcblk0      First SD/MMC card
                1 = /dev/mmcblk0p1    First partition on first MMC card
                8 = /dev/mmcblk1      Second SD/MMC card
                  ...

              The start of next SD/MMC card can be configured with
              CONFIG_MMC_BLOCK_MINORS, or overridden at boot/modprobe
```

```
              time using the mmcblk.perdev_minors option. That would
              bump the offset between each card to be the configured
              value instead of the default 8.

179 char      CCube DVXChip-based PCI products
                0 = /dev/dvxirq0      First DVX device
                1 = /dev/dvxirq1      Second DVX device
                  ...

180 char      USB devices
                0 = /dev/usb/lp0      First USB printer
                  ...
               15 = /dev/usb/lp15     16th USB printer
               48 = /dev/usb/scanner0 First USB scanner
                  ...
               63 = /dev/usb/scanner15 16th USB scanner
               64 = /dev/usb/rio500   Diamond Rio 500
               65 = /dev/usb/usblcd   USBLCD Interface (info@usblcd.de)
               66 = /dev/usb/cpad0    Synaptics cPad (mouse/LCD)
               96 = /dev/usb/hiddev0  1st USB HID device
                  ...
              111 = /dev/usb/hiddev15 16th USB HID device
              112 = /dev/usb/auer0    1st auerswald ISDN device
                  ...
              127 = /dev/usb/auer15   16th auerswald ISDN device
              128 = /dev/usb/brlvgr0  First Braille Voyager device
                  ...
              131 = /dev/usb/brlvgr3  Fourth Braille Voyager device
              132 = /dev/usb/idmouse  ID Mouse (fingerprint scanner) device
              133 = /dev/usb/sisusbvga1       First SiSUSB VGA device
                  ...
              140 = /dev/usb/sisusbvga8       Eighth SISUSB VGA device
              144 = /dev/usb/lcd      USB LCD device
              160 = /dev/usb/legousbtower0    1st USB Legotower device
                  ...
              175 = /dev/usb/legousbtower15   16th USB Legotower device
              176 = /dev/usb/usbtmc1  First USB TMC device
                 ...
              191 = /dev/usb/usbtmc16 16th USB TMC device
              192 = /dev/usb/yurex1   First USB Yurex device
                 ...
              209 = /dev/usb/yurex16  16th USB Yurex device

180 block     USB block devices
                0 = /dev/uba          First USB block device
                8 = /dev/ubb          Second USB block device
               16 = /dev/ubc          Third USB block device
                  ...

181 char      Conrad Electronic parallel port radio clocks
                0 = /dev/pcfclock0    First Conrad radio clock
                1 = /dev/pcfclock1    Second Conrad radio clock
                  ...

182 char      Picture Elements THR2 binarizer
                0 = /dev/pethr0       First THR2 board
                1 = /dev/pethr1       Second THR2 board
                  ...
```

```
183 char        SST 5136-DN DeviceNet interface
                  0 = /dev/ss5136dn0    First DeviceNet interface
                  1 = /dev/ss5136dn1    Second DeviceNet interface
                    ...

                This device used to be assigned to major number 144.
                It had to be moved due to an unfortunate conflict.

184 char        Picture Elements' video simulator/sender
                  0 = /dev/pevss0       First sender board
                  1 = /dev/pevss1       Second sender board
                    ...

185 char        InterMezzo high availability file system
                  0 = /dev/intermezzo0  First cache manager
                  1 = /dev/intermezzo1  Second cache manager
                    ...

                See http://web.archive.org/web/20080115195241/
                http://inter-mezzo.org/index.html

186 char        Object-based storage control device
                  0 = /dev/obd0         First obd control device
                  1 = /dev/obd1         Second obd control device
                    ...

                See ftp://ftp.lustre.org/pub/obd for code and information.

187 char        DESkey hardware encryption device
                  0 = /dev/deskey0      First DES key
                  1 = /dev/deskey1      Second DES key
                    ...

188 char        USB serial converters
                  0 = /dev/ttyUSB0      First USB serial converter
                  1 = /dev/ttyUSB1      Second USB serial converter
                    ...

189 char        USB serial converters - alternate devices
                  0 = /dev/cuusb0       Callout device for ttyUSB0
                  1 = /dev/cuusb1       Callout device for ttyUSB1
                    ...

190 char        Kansas City tracker/tuner card
                  0 = /dev/kctt0        First KCT/T card
                  1 = /dev/kctt1        Second KCT/T card
                    ...

191 char        Reserved for PCMCIA

192 char        Kernel profiling interface
                  0 = /dev/profile      Profiling control device
                  1 = /dev/profile0     Profiling device for CPU 0
                  2 = /dev/profile1     Profiling device for CPU 1
                    ...

193 char        Kernel event-tracing interface
```

```
                0 = /dev/trace        Tracing control device
                1 = /dev/trace0       Tracing device for CPU 0
                2 = /dev/trace1       Tracing device for CPU 1
                  ...

194 char        linVideoStreams (LINVS)
                0 = /dev/mvideo/status0       Video compression status
                1 = /dev/mvideo/stream0       Video stream
                2 = /dev/mvideo/frame0        Single compressed frame
                3 = /dev/mvideo/rawframe0     Raw uncompressed frame
                4 = /dev/mvideo/codec0        Direct codec access
                5 = /dev/mvideo/video4linux0  Video4Linux compatibility

               16 = /dev/mvideo/status1       Second device
                  ...
               32 = /dev/mvideo/status2       Third device
                  ...
                  ...
              240 = /dev/mvideo/status15      16th device
                  ...

195 char        Nvidia graphics devices
                0 = /dev/nvidia0              First Nvidia card
                1 = /dev/nvidia1              Second Nvidia card
                  ...
              255 = /dev/nvidiactl            Nvidia card control device

196 char        Tormenta T1 card
                0 = /dev/tor/0                Master control channel for all cards
                1 = /dev/tor/1                First DS0
                2 = /dev/tor/2                Second DS0
                  ...
               48 = /dev/tor/48               48th DS0
               49 = /dev/tor/49               First pseudo-channel
               50 = /dev/tor/50               Second pseudo-channel
                  ...

197 char        OpenTNF tracing facility
                0 = /dev/tnf/t0               Trace 0 data extraction
                1 = /dev/tnf/t1               Trace 1 data extraction
                  ...
              128 = /dev/tnf/status           Tracing facility status
              130 = /dev/tnf/trace            Tracing device

198 char        Total Impact TPMP2 quad coprocessor PCI card
                0 = /dev/tpmp2/0              First card
                1 = /dev/tpmp2/1              Second card
                  ...

199 char        Veritas volume manager (VxVM) volumes
                0 = /dev/vx/rdsk/*/*          First volume
                1 = /dev/vx/rdsk/*/*          Second volume
                  ...

199 block       Veritas volume manager (VxVM) volumes
                0 = /dev/vx/dsk/*/*           First volume
                1 = /dev/vx/dsk/*/*           Second volume
                  ...
```

```
              The namespace in these directories is maintained by
              the user space VxVM software.

200 char      Veritas VxVM configuration interface
                 0 = /dev/vx/config          Configuration access node
                 1 = /dev/vx/trace           Volume i/o trace access node
                 2 = /dev/vx/iod             Volume i/o daemon access node
                 3 = /dev/vx/info            Volume information access node
                 4 = /dev/vx/task            Volume tasks access node
                 5 = /dev/vx/taskmon         Volume tasks monitor daemon

201 char      Veritas VxVM dynamic multipathing driver
                0 = /dev/vx/rdmp/*           First multipath device
                1 = /dev/vx/rdmp/*           Second multipath device
                  ...
201 block     Veritas VxVM dynamic multipathing driver
                0 = /dev/vx/dmp/*            First multipath device
                1 = /dev/vx/dmp/*            Second multipath device
                  ...

              The namespace in these directories is maintained by
              the user space VxVM software.

202 char      CPU model-specific registers
                0 = /dev/cpu/0/msr           MSRs on CPU 0
                1 = /dev/cpu/1/msr           MSRs on CPU 1
                  ...

202 block     Xen Virtual Block Device
                0 = /dev/xvda       First Xen VBD whole disk
                16 = /dev/xvdb      Second Xen VBD whole disk
                32 = /dev/xvdc      Third Xen VBD whole disk
                  ...
                240 = /dev/xvdp     Sixteenth Xen VBD whole disk

              Partitions are handled in the same way as for IDE
              disks (see major number 3) except that the limit on
              partitions is 15.

203 char      CPU CPUID information
                0 = /dev/cpu/0/cpuid         CPUID on CPU 0
                1 = /dev/cpu/1/cpuid         CPUID on CPU 1
                  ...

204 char      Low-density serial ports
                0 = /dev/ttyLU0              LinkUp Systems L72xx UART - port 0
                1 = /dev/ttyLU1              LinkUp Systems L72xx UART - port 1
                2 = /dev/ttyLU2              LinkUp Systems L72xx UART - port 2
                3 = /dev/ttyLU3              LinkUp Systems L72xx UART - port 3
                4 = /dev/ttyFB0              Intel Footbridge (ARM)
                5 = /dev/ttySA0              StrongARM builtin serial port 0
                6 = /dev/ttySA1              StrongARM builtin serial port 1
                7 = /dev/ttySA2              StrongARM builtin serial port 2
                8 = /dev/ttySC0              SCI serial port (SuperH) - port 0
                9 = /dev/ttySC1              SCI serial port (SuperH) - port 1
               10 = /dev/ttySC2              SCI serial port (SuperH) - port 2
               11 = /dev/ttySC3              SCI serial port (SuperH) - port 3
```

```
                12 = /dev/ttyFW0              Firmware console - port 0
                13 = /dev/ttyFW1              Firmware console - port 1
                14 = /dev/ttyFW2              Firmware console - port 2
                15 = /dev/ttyFW3              Firmware console - port 3
                16 = /dev/ttyAM0              ARM ``AMBA'' serial port 0
                   ...
                31 = /dev/ttyAM15             ARM ``AMBA'' serial port 15
                32 = /dev/ttyDB0              DataBooster serial port 0
                   ...
                39 = /dev/ttyDB7              DataBooster serial port 7
                40 = /dev/ttySG0              SGI Altix console port
                41 = /dev/ttySMX0             Motorola i.MX - port 0
                42 = /dev/ttySMX1             Motorola i.MX - port 1
                43 = /dev/ttySMX2             Motorola i.MX - port 2
                44 = /dev/ttyMM0              Marvell MPSC - port 0
                45 = /dev/ttyMM1              Marvell MPSC - port 1
                46 = /dev/ttyCPM0             PPC CPM (SCC or SMC) - port 0
                   ...
                47 = /dev/ttyCPM5             PPC CPM (SCC or SMC) - port 5
                50 = /dev/ttyIOC0             Altix serial card
                   ...
                81 = /dev/ttyIOC31            Altix serial card
                82 = /dev/ttyVR0              NEC VR4100 series SIU
                83 = /dev/ttyVR1              NEC VR4100 series DSIU
                84 = /dev/ttyIOC84            Altix ioc4 serial card
                   ...
                115 = /dev/ttyIOC115          Altix ioc4 serial card
                116 = /dev/ttySIOC0           Altix ioc3 serial card
                   ...
                147 = /dev/ttySIOC31          Altix ioc3 serial card
                148 = /dev/ttyPSC0            PPC PSC - port 0
                   ...
                153 = /dev/ttyPSC5            PPC PSC - port 5
                154 = /dev/ttyAT0             ATMEL serial port 0
                   ...
                169 = /dev/ttyAT15            ATMEL serial port 15
                170 = /dev/ttyNX0             Hilscher netX serial port 0
                   ...
                185 = /dev/ttyNX15            Hilscher netX serial port 15
                186 = /dev/ttyJ0              JTAG1 DCC protocol based se-
rial port emulation
                187 = /dev/ttyUL0             Xilinx uartlite - port 0
                   ...
                190 = /dev/ttyUL3             Xilinx uartlite - port 3
                191 = /dev/xvc0               Xen virtual console - port 0
                192 = /dev/ttyPZ0             pmac_zilog - port 0
                   ...
                195 = /dev/ttyPZ3             pmac_zilog - port 3
                196 = /dev/ttyTX0             TX39/49 serial port 0
                   ...
                204 = /dev/ttyTX7             TX39/49 serial port 7
                205 = /dev/ttySC0             SC26xx serial port 0
                206 = /dev/ttySC1             SC26xx serial port 1
                207 = /dev/ttySC2             SC26xx serial port 2
                208 = /dev/ttySC3             SC26xx serial port 3
                209 = /dev/ttyMAX0            MAX3100 serial port 0
                210 = /dev/ttyMAX1            MAX3100 serial port 1
                211 = /dev/ttyMAX2            MAX3100 serial port 2
```

```
              212 = /dev/ttyMAX3            MAX3100 serial port 3

205 char      Low-density serial ports (alternate device)
                0 = /dev/culu0              Callout device for ttyLU0
                1 = /dev/culu1              Callout device for ttyLU1
                2 = /dev/culu2              Callout device for ttyLU2
                3 = /dev/culu3              Callout device for ttyLU3
                4 = /dev/cufb0              Callout device for ttyFB0
                5 = /dev/cusa0              Callout device for ttySA0
                6 = /dev/cusa1              Callout device for ttySA1
                7 = /dev/cusa2              Callout device for ttySA2
                8 = /dev/cusc0              Callout device for ttySC0
                9 = /dev/cusc1              Callout device for ttySC1
               10 = /dev/cusc2              Callout device for ttySC2
               11 = /dev/cusc3              Callout device for ttySC3
               12 = /dev/cufw0              Callout device for ttyFW0
               13 = /dev/cufw1              Callout device for ttyFW1
               14 = /dev/cufw2              Callout device for ttyFW2
               15 = /dev/cufw3              Callout device for ttyFW3
               16 = /dev/cuam0              Callout device for ttyAM0
                  ...
               31 = /dev/cuam15             Callout device for ttyAM15
               32 = /dev/cudb0              Callout device for ttyDB0
                  ...
               39 = /dev/cudb7              Callout device for ttyDB7
               40 = /dev/cusg0              Callout device for ttySG0
               41 = /dev/ttycusmx0          Callout device for ttySMX0
               42 = /dev/ttycusmx1          Callout device for ttySMX1
               43 = /dev/ttycusmx2          Callout device for ttySMX2
               46 = /dev/cucpm0             Callout device for ttyCPM0
                  ...
               49 = /dev/cucpm5             Callout device for ttyCPM5
               50 = /dev/cuioc40            Callout device for ttyIOC40
                  ...
               81 = /dev/cuioc431           Callout device for ttyIOC431
               82 = /dev/cuvr0              Callout device for ttyVR0
               83 = /dev/cuvr1              Callout device for ttyVR1

206 char      OnStream SC-x0 tape devices
                0 = /dev/osst0              First OnStream SCSI tape, mode 0
                1 = /dev/osst1              Second OnStream SCSI tape, mode 0
                  ...
               32 = /dev/osst0l             First OnStream SCSI tape, mode 1
               33 = /dev/osst1l             Second OnStream SCSI tape, mode 1
                  ...
               64 = /dev/osst0m             First OnStream SCSI tape, mode 2
               65 = /dev/osst1m             Second OnStream SCSI tape, mode 2
                  ...
               96 = /dev/osst0a             First OnStream SCSI tape, mode 3
               97 = /dev/osst1a             Second OnStream SCSI tape, mode 3
                  ...
              128 = /dev/nosst0             No rewind version of /dev/osst0
              129 = /dev/nosst1             No rewind version of /dev/osst1
                  ...
              160 = /dev/nosst0l            No rewind version of /dev/osst0l
              161 = /dev/nosst1l            No rewind version of /dev/osst1l
                  ...
              192 = /dev/nosst0m            No rewind version of /dev/osst0m
```

```
                193 = /dev/nosst1m              No rewind version of /dev/osst1m
                    ...
                224 = /dev/nosst0a              No rewind version of /dev/osst0a
                225 = /dev/nosst1a              No rewind version of /dev/osst1a
                    ...

                The OnStream SC-x0 SCSI tapes do not support the
                standard SCSI SASD command set and therefore need
                their own driver ``osst''. Note that the IDE, USB (and
                maybe ParPort) versions may be driven via ide-scsi or
                usb-storage SCSI emulation and this osst device and
                driver as well.  The ADR-x0 drives are QIC-157
                compliant and don't need osst.

207 char        Compaq ProLiant health feature indicate
                  0 = /dev/cpqhealth/cpqw       Redirector interface
                  1 = /dev/cpqhealth/crom       EISA CROM
                  2 = /dev/cpqhealth/cdt        Data Table
                  3 = /dev/cpqhealth/cevt       Event Log
                  4 = /dev/cpqhealth/casr       Automatic Server Recovery
                  5 = /dev/cpqhealth/cecc       ECC Memory
                  6 = /dev/cpqhealth/cmca       Machine Check Architecture
                  7 = /dev/cpqhealth/ccsm       Deprecated CDT
                  8 = /dev/cpqhealth/cnmi       NMI Handling
                  9 = /dev/cpqhealth/css        Sideshow Management
                 10 = /dev/cpqhealth/cram       CMOS interface
                 11 = /dev/cpqhealth/cpci       PCI IRQ interface

208 char        User space serial ports
                  0 = /dev/ttyU0                First user space serial port
                  1 = /dev/ttyU1                Second user space serial port
                    ...

209 char        User space serial ports (alternate devices)
                  0 = /dev/cuu0                 Callout device for ttyU0
                  1 = /dev/cuu1                 Callout device for ttyU1
                    ...

210 char        SBE, Inc. sync/async serial card
                  0 = /dev/sbei/wxcfg0          Configuration device for board 0
                  1 = /dev/sbei/dld0            Download device for board 0
                  2 = /dev/sbei/wan00           WAN device, port 0, board 0
                  3 = /dev/sbei/wan01           WAN device, port 1, board 0
                  4 = /dev/sbei/wan02           WAN device, port 2, board 0
                  5 = /dev/sbei/wan03           WAN device, port 3, board 0
                  6 = /dev/sbei/wanc00          WAN clone device, port 0, board 0
                  7 = /dev/sbei/wanc01          WAN clone device, port 1, board 0
                  8 = /dev/sbei/wanc02          WAN clone device, port 2, board 0
                  9 = /dev/sbei/wanc03          WAN clone device, port 3, board 0
                 10 = /dev/sbei/wxcfg1          Configuration device for board 1
                 11 = /dev/sbei/dld1            Download device for board 1
                 12 = /dev/sbei/wan10           WAN device, port 0, board 1
                 13 = /dev/sbei/wan11           WAN device, port 1, board 1
                 14 = /dev/sbei/wan12           WAN device, port 2, board 1
                 15 = /dev/sbei/wan13           WAN device, port 3, board 1
                 16 = /dev/sbei/wanc10          WAN clone device, port 0, board 1
                 17 = /dev/sbei/wanc11          WAN clone device, port 1, board 1
                 18 = /dev/sbei/wanc12          WAN clone device, port 2, board 1
```

```
                 19 = /dev/sbei/wanc13          WAN clone device, port 3, board 1
                    ...

                Yes, each board is really spaced 10 (decimal) apart.

 211 char       Addinum CPCI1500 digital I/O card
                  0 = /dev/addinum/cpci1500/0   First CPCI1500 card
                  1 = /dev/addinum/cpci1500/1   Second CPCI1500 card
                    ...

 212 char       LinuxTV.org DVB driver subsystem
                  0 = /dev/dvb/adapter0/video0    first video decoder of first card
                  1 = /dev/dvb/adapter0/audio0    first audio decoder of first card
                  2 = /dev/dvb/adapter0/sec0      (obsolete/unused)
                  3 = /dev/dvb/adapter0/frontend0 first frontend device of first card
                  4 = /dev/dvb/adapter0/demux0    first demux device of first card
                  5 = /dev/dvb/adapter0/dvr0      first digital video recoder de-
vice of first card
                  6 = /dev/dvb/adapter0/ca0       first common ac-
cess port of first card
                  7 = /dev/dvb/adapter0/net0      first network device of first card
                  8 = /dev/dvb/adapter0/osd0      first on-screen-display de-
vice of first card
                  9 = /dev/dvb/adapter0/video1    second video decoder of first card
                    ...
                 64 = /dev/dvb/adapter1/video0    first video decoder of second card
                    ...
                128 = /dev/dvb/adapter2/video0    first video decoder of third card
                    ...
                196 = /dev/dvb/adapter3/video0    first video decoder of fourth card

 216 char       Bluetooth RFCOMM TTY devices
                  0 = /dev/rfcomm0              First Bluetooth RFCOMM TTY device
                  1 = /dev/rfcomm1              Second Bluetooth RFCOMM TTY device
                    ...

 217 char       Bluetooth RFCOMM TTY devices (alternate devices)
                  0 = /dev/curf0                Callout device for rfcomm0
                  1 = /dev/curf1                Callout device for rfcomm1
                    ...

 218 char       The Logical Company bus Unibus/Qbus adapters
                  0 = /dev/logicalco/bci/0      First bus adapter
                  1 = /dev/logicalco/bci/1      First bus adapter
                    ...

 219 char       The Logical Company DCI-1300 digital I/O card
                  0 = /dev/logicalco/dci1300/0  First DCI-1300 card
                  1 = /dev/logicalco/dci1300/1  Second DCI-1300 card
                    ...

 220 char       Myricom Myrinet ``GM'' board
                  0 = /dev/myricom/gm0          First Myrinet GM board
                  1 = /dev/myricom/gmp0         First board ``root access''
                  2 = /dev/myricom/gm1          Second Myrinet GM board
                  3 = /dev/myricom/gmp1         Second board ``root access''
                    ...
```

```
221 char        VME bus
                  0 = /dev/bus/vme/m0           First master image
                  1 = /dev/bus/vme/m1           Second master image
                  2 = /dev/bus/vme/m2           Third master image
                  3 = /dev/bus/vme/m3           Fourth master image
                  4 = /dev/bus/vme/s0           First slave image
                  5 = /dev/bus/vme/s1           Second slave image
                  6 = /dev/bus/vme/s2           Third slave image
                  7 = /dev/bus/vme/s3           Fourth slave image
                  8 = /dev/bus/vme/ctl          Control

                It is expected that all VME bus drivers will use the
                same interface.  For interface documentation see
                http://www.vmelinux.org/.

224 char        A2232 serial card
                  0 = /dev/ttyY0                First A2232 port
                  1 = /dev/ttyY1                Second A2232 port
                    ...

225 char        A2232 serial card (alternate devices)
                  0 = /dev/cuy0                 Callout device for ttyY0
                  1 = /dev/cuy1                 Callout device for ttyY1
                    ...

226 char        Direct Rendering Infrastructure (DRI)
                  0 = /dev/dri/card0            First graphics card
                  1 = /dev/dri/card1            Second graphics card
                    ...

227 char        IBM 3270 terminal Unix tty access
                  1 = /dev/3270/tty1            First 3270 terminal
                  2 = /dev/3270/tty2            Seconds 3270 terminal
                    ...

228 char        IBM 3270 terminal block-mode access
                  0 = /dev/3270/tub             Controlling interface
                  1 = /dev/3270/tub1            First 3270 terminal
                  2 = /dev/3270/tub2            Second 3270 terminal
                    ...

229 char        IBM iSeries/pSeries virtual console
                  0 = /dev/hvc0                 First console port
                  1 = /dev/hvc1                 Second console port
                    ...

230 char        IBM iSeries virtual tape
                  0 = /dev/iseries/vt0          First virtual tape, mode 0
                  1 = /dev/iseries/vt1          Second virtual tape, mode 0
                    ...
                 32 = /dev/iseries/vt0l         First virtual tape, mode 1
                 33 = /dev/iseries/vt1l         Second virtual tape, mode 1
                    ...
                 64 = /dev/iseries/vt0m         First virtual tape, mode 2
                 65 = /dev/iseries/vt1m         Second virtual tape, mode 2
                    ...
                 96 = /dev/iseries/vt0a         First virtual tape, mode 3
                 97 = /dev/iseries/vt1a         Second virtual tape, mode 3
```

```
                  ...
                128 = /dev/iseries/nvt0         First virtual tape, mode 0, no rewind
                129 = /dev/iseries/nvt1         Second virtual tape, mode 0, no rewind
                    ...
                160 = /dev/iseries/nvt0l        First virtual tape, mode 1, no rewind
                161 = /dev/iseries/nvt1l        Second virtual tape, mode 1, no rewind
                    ...
                192 = /dev/iseries/nvt0m        First virtual tape, mode 2, no rewind
                193 = /dev/iseries/nvt1m        Second virtual tape, mode 2, no rewind
                    ...
                224 = /dev/iseries/nvt0a        First virtual tape, mode 3, no rewind
                225 = /dev/iseries/nvt1a        Second virtual tape, mode 3, no rewind
                    ...

                ``No rewind'' refers to the omission of the default
                automatic rewind on device close.  The MTREW or MTOFFL
                ioctl()'s can be used to rewind the tape regardless of
                the device used to access it.

 231 char       InfiniBand
                0 = /dev/infiniband/umad0
                1 = /dev/infiniband/umad1
                  ...
                63 = /dev/infiniband/umad63    63rd InfiniBandMad device
                64 = /dev/infiniband/issm0     First InfiniBand IsSM device
                65 = /dev/infiniband/issm1     Second InfiniBand IsSM device
                  ...
                127 = /dev/infiniband/issm63    63rd InfiniBand IsSM device
                128 = /dev/infiniband/uverbs0   First InfiniBand verbs device
                129 = /dev/infiniband/uverbs1   Second InfiniBand verbs device
                  ...
                159 = /dev/infiniband/uverbs31  31st InfiniBand verbs device

 232 char       Biometric Devices
                0 = /dev/biometric/sensor0/fingerprint  first fingerprint sen-
sor on first device
                1 = /dev/biometric/sensor0/iris         first iris sen-
sor on first device
                2 = /dev/biometric/sensor0/retina       first retina sen-
sor on first device
                3 = /dev/biometric/sensor0/voiceprint   first voiceprint sen-
sor on first device
                4 = /dev/biometric/sensor0/facial       first facial sen-
sor on first device
                5 = /dev/biometric/sensor0/hand         first hand sen-
sor on first device
                  ...
                10 = /dev/biometric/sensor1/fingerprint first fingerprint sen-
sor on second device
                  ...
                20 = /dev/biometric/sensor2/fingerprint first fingerprint sen-
sor on third device
                  ...

 233 char       PathScale InfiniPath interconnect
                0 = /dev/ipath        Primary device for programs (any unit)
                1 = /dev/ipath0       Access specifically to unit 0
                2 = /dev/ipath1       Access specifically to unit 1
```

```
                ...
               4 = /dev/ipath3       Access specifically to unit 3
               129 = /dev/ipath_sma    Device used by Subnet Management Agent
               130 = /dev/ipath_diag   Device used by diagnostics programs

 234-254       char    RESERVED FOR DYNAMIC ASSIGNMENT
               Character devices that request a dynamic allocation of major num-
ber will
               take numbers starting from 254 and downward.

 240-254 block  LOCAL/EXPERIMENTAL USE
               Allocated for local/experimental use.  For devices not
               assigned official numbers, these ranges should be
               used in order to avoid conflicting with future assignments.

 255 char      RESERVED

 255 block     RESERVED

               This major is reserved to assist the expansion to a
               larger number space.  No device nodes with this major
               should ever be created on the filesystem.
               (This is probably not true anymore, but I'll leave it
               for now /Torben)

 ---LARGE MAJORS!!!!!---

 256 char      Equinox SST multi-port serial boards
                  0 = /dev/ttyEQ0      First serial port on first Equinox SST board
                127 = /dev/ttyEQ127    Last serial port on first Equinox SST board
                128 = /dev/ttyEQ128    First serial port on second Equinox SST board
                ...
               1027 = /dev/ttyEQ1027   Last serial port on eighth Equinox SST board

 256 block     Resident Flash Disk Flash Translation Layer
                 0 = /dev/rfda         First RFD FTL layer
                16 = /dev/rfdb         Second RFD FTL layer
                ...
               240 = /dev/rfdp         16th RFD FTL layer

 257 char      Phoenix Technologies Cryptographic Services Driver
                 0 = /dev/ptlsec       Crypto Services Driver

 257 block     SSFDC Flash Translation Layer filesystem
                 0 = /dev/ssfdca       First SSFDC layer
                 8 = /dev/ssfdcb       Second SSFDC layer
                16 = /dev/ssfdcc       Third SSFDC layer
                24 = /dev/ssfdcd       4th SSFDC layer
                32 = /dev/ssfdce       5th SSFDC layer
                40 = /dev/ssfdcf       6th SSFDC layer
                48 = /dev/ssfdcg       7th SSFDC layer
                56 = /dev/ssfdch       8th SSFDC layer

 258 block     ROM/Flash read-only translation layer
                 0 = /dev/blockrom0    First ROM card's translation layer interface
                 1 = /dev/blockrom1    Second ROM card's translation layer interface
                 ...
```

```
259 block       Block Extended Major
                  Used dynamically to hold additional partition minor
                  numbers and allow large numbers of partitions per device

259 char        FPGA configuration interfaces
                  0 = /dev/icap0        First Xilinx internal configuration
                  1 = /dev/icap1        Second Xilinx internal configuration

260 char        OSD (Object-based-device) SCSI Device
                  0 = /dev/osd0         First OSD Device
                  1 = /dev/osd1         Second OSD Device
                  ...
                  255 = /dev/osd255     256th OSD Device

384-511 char    RESERVED FOR DYNAMIC ASSIGNMENT
                Character devices that request a dynamic allocation of major
                number will take numbers starting from 511 and downward,
                once the 234-254 range is full.
```

Additional /dev/ directory entries

This section details additional entries that should or may exist in the /dev directory. It is preferred that symbolic links use the same form (absolute or relative) as is indicated here. Links are classified as “hard” or “symbolic” depending on the preferred type of link; if possible, the indicated type of link should be used.

Compulsory links

These links should exist on all systems:

/dev/fd	/proc/self/fd	symbolic	File descriptors
/dev/stdin	fd/0	symbolic	stdin file descriptor
/dev/stdout	fd/1	symbolic	stdout file descriptor
/dev/stderr	fd/2	symbolic	stderr file descriptor
/dev/nfsd	socksys	symbolic	Required by iBCS-2
/dev/X0R	null	symbolic	Required by iBCS-2

Note: /dev/X0R is <letter X>-<digit 0>-<letter R>.

Recommended links

It is recommended that these links exist on all systems:

/dev/core	/proc/kcore	symbolic	Backward compatibility
/dev/ramdisk	ram0	symbolic	Backward compatibility
/dev/ftape	qft0	symbolic	Backward compatibility
/dev/bttv0	video0	symbolic	Backward compatibility
/dev/radio	radio0	symbolic	Backward compatibility
/dev/i2o*	/dev/i2o/*	symbolic	Backward compatibility
/dev/scd?	sr?	hard	Alternate SCSI CD-ROM name

Locally defined links

The following links may be established locally to conform to the configuration of the system. This is merely a tabulation of existing practice, and does not constitute a recommendation. However, if they exist, they should have the following uses.

/dev/mouse	mouse port	symbolic	Current mouse device
/dev/tape	tape device	symbolic	Current tape device
/dev/cdrom	CD-ROM device	symbolic	Current CD-ROM device
/dev/cdwriter	CD-writer	symbolic	Current CD-writer device
/dev/scanner	scanner	symbolic	Current scanner device
/dev/modem	modem port	symbolic	Current dialout device
/dev/root	root device	symbolic	Current root filesystem
/dev/swap	swap device	symbolic	Current swap device

`/dev/modem` should not be used for a modem which supports dialin as well as dialout, as it tends to cause lock file problems. If it exists, `/dev/modem` should point to the appropriate primary TTY device (the use of the alternate callout devices is deprecated).

For SCSI devices, `/dev/tape` and `/dev/cdrom` should point to the *cooked* devices (`/dev/st*` and `/dev/sr*`, respectively), whereas `/dev/cdwriter` and /dev/scanner should point to the appropriate generic SCSI devices (/dev/sg*).

`/dev/mouse` may point to a primary serial TTY device, a hardware mouse device, or a socket for a mouse driver program (e.g. `/dev/gpmdata`).

Sockets and pipes

Non-transient sockets and named pipes may exist in /dev. Common entries are:

/dev/printer	socket	lpd local socket
/dev/log	socket	syslog local socket
/dev/gpmdata	socket	gpm mouse multiplexer

Mount points

The following names are reserved for mounting special filesystems under /dev. These special filesystems provide kernel interfaces that cannot be provided with standard device nodes.

/dev/pts	devpts	PTY slave filesystem
/dev/shm	tmpfs	POSIX shared memory maintenance access

Terminal devices

Terminal, or TTY devices are a special class of character devices. A terminal device is any device that could act as a controlling terminal for a session; this includes virtual consoles, serial ports, and pseudoterminals (PTYs).

All terminal devices share a common set of capabilities known as line disciplines; these include the common terminal line discipline as well as SLIP and PPP modes.

All terminal devices are named similarly; this section explains the naming and use of the various types of TTYs. Note that the naming conventions include several historical warts; some of these are Linux-specific, some were inherited from other systems, and some reflect Linux outgrowing a borrowed convention.

A hash mark (#) in a device name is used here to indicate a decimal number without leading zeroes.

Virtual consoles and the console device

Virtual consoles are full-screen terminal displays on the system video monitor. Virtual consoles are named /dev/tty#, with numbering starting at /dev/tty1; /dev/tty0 is the current virtual console. /dev/tty0 is the device that should be used to access the system video card on those architectures for which the frame buffer devices (/dev/fb*) are not applicable. Do not use /dev/console for this purpose.

The console device, /dev/console, is the device to which system messages should be sent, and on which logins should be permitted in single-user mode. Starting with Linux 2.1.71, /dev/console is managed by the kernel; for previous versions it should be a symbolic link to either /dev/tty0, a specific virtual console such as /dev/tty1, or to a serial port primary (tty*, not cu*) device, depending on the configuration of the system.

Serial ports

Serial ports are RS-232 serial ports and any device which simulates one, either in hardware (such as internal modems) or in software (such as the ISDN driver.) Under Linux, each serial ports has two device names, the primary or callin device and the alternate or callout one. Each kind of device is indicated by a different letter. For any letter X, the names of the devices are /dev/ttyX# and /dev/cux#, respectively; for historical reasons, /dev/ttyS# and /dev/ttyC# correspond to /dev/cua# and /dev/cub#. In the future, it should be expected that multiple letters will be used; all letters will be upper case for the "tty" device (e.g. /dev/ttyDP#) and lower case for the "cu" device (e.g. /dev/cudp#).

The names /dev/ttyQ# and /dev/cuq# are reserved for local use.

The alternate devices provide for kernel-based exclusion and somewhat different defaults than the primary devices. Their main purpose is to allow the use of serial ports with programs with no inherent or broken support for serial ports. Their use is deprecated, and they may be removed from a future version of Linux.

Arbitration of serial ports is provided by the use of lock files with the names /var/lock/LCK..ttyX#. The contents of the lock file should be the PID of the locking process as an ASCII number.

It is common practice to install links such as /dev/modem which point to serial ports. In order to ensure proper locking in the presence of these links, it is recommended that software chase symlinks and lock all possible names; additionally, it is recommended that a lock file be installed with the corresponding alternate device. In order to avoid deadlocks, it is recommended that the locks are acquired in the following order, and released in the reverse:

1. The symbolic link name, if any (/var/lock/LCK..modem)
2. The "tty" name (/var/lock/LCK..ttyS2)
3. The alternate device name (/var/lock/LCK..cua2)

In the case of nested symbolic links, the lock files should be installed in the order the symlinks are resolved.

Under no circumstances should an application hold a lock while waiting for another to be released. In addition, applications which attempt to create lock files for the corresponding alternate device names should take into account the possibility of being used on a non-serial port TTY, for which no alternate device would exist.

Pseudoterminals (PTYs)

Pseudoterminals, or PTYs, are used to create login sessions or provide other capabilities requiring a TTY line discipline (including SLIP or PPP capability) to arbitrary data-generation processes. Each PTY has a master side, named /dev/pty[p-za-e][0-9a-f], and a slave side, named /dev/tty[p-za-e][0-9a-f]. The kernel arbitrates the use of PTYs by allowing each master side to be opened only once.

Once the master side has been opened, the corresponding slave device can be used in the same manner as any TTY device. The master and slave devices are connected by the kernel, generating the equivalent of a bidirectional pipe with TTY capabilities.

Recent versions of the Linux kernels and GNU libc contain support for the System V/Unix98 naming scheme for PTYs, which assigns a common device, `/dev/ptmx`, to all the masters (opening it will automatically give you a previously unassigned PTY) and a subdirectory, `/dev/pts`, for the slaves; the slaves are named with decimal integers (`/dev/pts/#` in our notation). This removes the problem of exhausting the namespace and enables the kernel to automatically create the device nodes for the slaves on demand using the "devpts" filesystem.

Here is a set of documents aimed at users who are trying to track down problems and bugs in particular.

CHAPTER

FOUR

REPORTING BUGS

Background

The upstream Linux kernel maintainers only fix bugs for specific kernel versions. Those versions include the current “release candidate” (or -rc) kernel, any “stable” kernel versions, and any “long term” kernels.

Please see https://www.kernel.org/ for a list of supported kernels. Any kernel marked with [EOL] is “end of life” and will not have any fixes backported to it.

If you’ve found a bug on a kernel version that isn’t listed on kernel.org, contact your Linux distribution or embedded vendor for support. Alternatively, you can attempt to run one of the supported stable or -rc kernels, and see if you can reproduce the bug on that. It’s preferable to reproduce the bug on the latest -rc kernel.

How to report Linux kernel bugs

Identify the problematic subsystem

Identifying which part of the Linux kernel might be causing your issue increases your chances of getting your bug fixed. Simply posting to the generic linux-kernel mailing list (LKML) may cause your bug report to be lost in the noise of a mailing list that gets 1000+ emails a day.

Instead, try to figure out which kernel subsystem is causing the issue, and email that subsystem’s maintainer and mailing list. If the subsystem maintainer doesn’t answer, then expand your scope to mailing lists like LKML.

Identify who to notify

Once you know the subsystem that is causing the issue, you should send a bug report. Some maintainers prefer bugs to be reported via bugzilla (https://bugzilla.kernel.org), while others prefer that bugs be reported via the subsystem mailing list.

To find out where to send an emailed bug report, find your subsystem or device driver in the MAINTAINERS file. Search in the file for relevant entries, and send your bug report to the person(s) listed in the “M:” lines, making sure to Cc the mailing list(s) in the “L:” lines. When the maintainer replies to you, make sure to ‘Reply-all’ in order to keep the public mailing list(s) in the email thread.

If you know which driver is causing issues, you can pass one of the driver files to the get_maintainer.pl script:

```
perl scripts/get_maintainer.pl -f <filename>
```

If it is a security bug, please copy the Security Contact listed in the MAINTAINERS file. They can help coordinate bugfix and disclosure. See *Documentation/admin-guide/security-bugs.rst* for more information.

If you can't figure out which subsystem caused the issue, you should file a bug in kernel.org bugzilla and send email to linux-kernel@vger.kernel.org, referencing the bugzilla URL. (For more information on the linux-kernel mailing list see http://www.tux.org/lkml/).

Tips for reporting bugs

If you haven't reported a bug before, please read:

http://www.chiark.greenend.org.uk/~sgtatham/bugs.html

http://www.catb.org/esr/faqs/smart-questions.html

It's REALLY important to report bugs that seem unrelated as separate email threads or separate bugzilla entries. If you report several unrelated bugs at once, it's difficult for maintainers to tease apart the relevant data.

Gather information

The most important information in a bug report is how to reproduce the bug. This includes system information, and (most importantly) step-by-step instructions for how a user can trigger the bug.

If the failure includes an "OOPS:", take a picture of the screen, capture a netconsole trace, or type the message from your screen into the bug report. Please read "Documentation/admin-guide/oops-tracing.rst" before posting your bug report. This explains what you should do with the "Oops" information to make it useful to the recipient.

This is a suggested format for a bug report sent via email or bugzilla. Having a standardized bug report form makes it easier for you not to overlook things, and easier for the developers to find the pieces of information they're really interested in. If some information is not relevant to your bug, feel free to exclude it.

First run the ver_linux script included as scripts/ver_linux, which reports the version of some important subsystems. Run this script with the command `awk -f scripts/ver_linux`.

Use that information to fill in all fields of the bug report form, and post it to the mailing list with a subject of "PROBLEM: <one line summary from [1.]>" for easy identification by the developers:

```
[1.] One line summary of the problem:
[2.] Full description of the problem/report:
[3.] Keywords (i.e., modules, networking, kernel):
[4.] Kernel information
[4.1.] Kernel version (from /proc/version):
[4.2.] Kernel .config file:
[5.] Most recent kernel version which did not have the bug:
[6.] Output of Oops.. message (if applicable) with symbolic information
     resolved (see Documentation/admin-guide/oops-tracing.rst)
[7.] A small shell script or example program which triggers the
     problem (if possible)
[8.] Environment
[8.1.] Software (add the output of the ver_linux script here)
[8.2.] Processor information (from /proc/cpuinfo):
[8.3.] Module information (from /proc/modules):
[8.4.] Loaded driver and hardware information (/proc/ioports, /proc/iomem)
[8.5.] PCI information ('lspci -vvv' as root)
[8.6.] SCSI information (from /proc/scsi/scsi)
[8.7.] Other information that might be relevant to the problem
       (please look in /proc and include all information that you
       think to be relevant):
[X.] Other notes, patches, fixes, workarounds:
```

Follow up

Expectations for bug reporters

Linux kernel maintainers expect bug reporters to be able to follow up on bug reports. That may include running new tests, applying patches, recompiling your kernel, and/or re-triggering your bug. The most frustrating thing for maintainers is for someone to report a bug, and then never follow up on a request to try out a fix.

That said, it's still useful for a kernel maintainer to know a bug exists on a supported kernel, even if you can't follow up with retests. Follow up reports, such as replying to the email thread with "I tried the latest kernel and I can't reproduce my bug anymore" are also helpful, because maintainers have to assume silence means things are still broken.

Expectations for kernel maintainers

Linux kernel maintainers are busy, overworked human beings. Some times they may not be able to address your bug in a day, a week, or two weeks. If they don't answer your email, they may be on vacation, or at a Linux conference. Check the conference schedule at https://LWN.net for more info:

> https://lwn.net/Calendar/

In general, kernel maintainers take 1 to 5 business days to respond to bugs. The majority of kernel maintainers are employed to work on the kernel, and they may not work on the weekends. Maintainers are scattered around the world, and they may not work in your time zone. Unless you have a high priority bug, please wait at least a week after the first bug report before sending the maintainer a reminder email.

The exceptions to this rule are regressions, kernel crashes, security holes, or userspace breakage caused by new kernel behavior. Those bugs should be addressed by the maintainers ASAP. If you suspect a maintainer is not responding to these types of bugs in a timely manner (especially during a merge window), escalate the bug to LKML and Linus Torvalds.

Thank you!

[Some of this is taken from Frohwalt Egerer's original linux-kernel FAQ]

CHAPTER

FIVE

SECURITY BUGS

Linux kernel developers take security very seriously. As such, we'd like to know when a security bug is found so that it can be fixed and disclosed as quickly as possible. Please report security bugs to the Linux kernel security team.

Contact

The Linux kernel security team can be contacted by email at <security@kernel.org>. This is a private list of security officers who will help verify the bug report and develop and release a fix. If you already have a fix, please include it with your report, as that can speed up the process considerably. It is possible that the security team will bring in extra help from area maintainers to understand and fix the security vulnerability.

As it is with any bug, the more information provided the easier it will be to diagnose and fix. Please review the procedure outlined in admin-guide/reporting-bugs.rst if you are unclear about what information is helpful. Any exploit code is very helpful and will not be released without consent from the reporter unless it has already been made public.

Disclosure

The goal of the Linux kernel security team is to work with the bug submitter to bug resolution as well as disclosure. We prefer to fully disclose the bug as soon as possible. It is reasonable to delay disclosure when the bug or the fix is not yet fully understood, the solution is not well-tested or for vendor coordination. However, we expect these delays to be short, measurable in days, not weeks or months. A disclosure date is negotiated by the security team working with the bug submitter as well as vendors. However, the kernel security team holds the final say when setting a disclosure date. The timeframe for disclosure is from immediate (esp. if it's already publicly known) to a few weeks. As a basic default policy, we expect report date to disclosure date to be on the order of 7 days.

Coordination

Fixes for sensitive bugs, such as those that might lead to privilege escalations, may need to be coordinated with the private <linux-distros@vs.openwall.org> mailing list so that distribution vendors are well prepared to issue a fixed kernel upon public disclosure of the upstream fix. Distros will need some time to test the proposed patch and will generally request at least a few days of embargo, and vendor update publication prefers to happen Tuesday through Thursday. When appropriate, the security team can assist with this coordination, or the reporter can include linux-distros from the start. In this case, remember to prefix the email Subject line with "[vs]" as described in the linux-distros wiki: <http://oss-security.openwall.org/wiki/mailing-lists/distros#how-to-use-the-lists>

CVE assignment

The security team does not normally assign CVEs, nor do we require them for reports or fixes, as this can needlessly complicate the process and may delay the bug handling. If a reporter wishes to have a CVE identifier assigned ahead of public disclosure, they will need to contact the private linux-distros list, described above. When such a CVE identifier is known before a patch is provided, it is desirable to mention it in the commit message, though.

Non-disclosure agreements

The Linux kernel security team is not a formal body and therefore unable to enter any non-disclosure agreements.

CHAPTER

SIX

BUG HUNTING

Kernel bug reports often come with a stack dump like the one below:

```
------------[ cut here ]------------
WARNING: CPU: 1 PID: 28102 at kernel/module.c:1108 module_put+0x57/0x70
Modules linked in: dvb_usb_gp8psk(-) dvb_usb dvb_core nvidia_drm(PO) nvidia_modeset(PO) snd_hda_
 ↪codec_hdmi snd_hda_intel snd_hda_codec snd_hwdep snd_hda_core snd_pcm snd_timer snd soundcore␣
 ↪nvidia(PO) [last unloaded: rc_core]
CPU: 1 PID: 28102 Comm: rmmod Tainted: P        WC O 4.8.4-build.1 #1
Hardware name: MSI MS-7309/MS-7309, BIOS V1.12 02/23/2009
 00000000 c12ba080 00000000 00000000 c103ed6a c1616014 00000001 00006dc6
 c1615862 00000454 c109e8a7 c109e8a7 00000009 ffffffff 00000000 f13f6a10
 f5f5a600 c103ee33 00000009 00000000 00000000 c109e8a7 f80ca4d0 c109f617
Call Trace:
 [<c12ba080>] ? dump_stack+0x44/0x64
 [<c103ed6a>] ? __warn+0xfa/0x120
 [<c109e8a7>] ? module_put+0x57/0x70
 [<c109e8a7>] ? module_put+0x57/0x70
 [<c103ee33>] ? warn_slowpath_null+0x23/0x30
 [<c109e8a7>] ? module_put+0x57/0x70
 [<f80ca4d0>] ? gp8psk_fe_set_frontend+0x460/0x460 [dvb_usb_gp8psk]
 [<c109f617>] ? symbol_put_addr+0x27/0x50
 [<f80bc9ca>] ? dvb_usb_adapter_frontend_exit+0x3a/0x70 [dvb_usb]
 [<f80bb3bf>] ? dvb_usb_exit+0x2f/0xd0 [dvb_usb]
 [<c13d03bc>] ? usb_disable_endpoint+0x7c/0xb0
 [<f80bb48a>] ? dvb_usb_device_exit+0x2a/0x50 [dvb_usb]
 [<c13d2882>] ? usb_unbind_interface+0x62/0x250
 [<c136b514>] ? __pm_runtime_idle+0x44/0x70
 [<c13620d8>] ? __device_release_driver+0x78/0x120
 [<c1362907>] ? driver_detach+0x87/0x90
 [<c1361c48>] ? bus_remove_driver+0x38/0x90
 [<c13d1c18>] ? usb_deregister+0x58/0xb0
 [<c109fbb0>] ? SyS_delete_module+0x130/0x1f0
 [<c1055654>] ? task_work_run+0x64/0x80
 [<c1000fa5>] ? exit_to_usermode_loop+0x85/0x90
 [<c10013f0>] ? do_fast_syscall_32+0x80/0x130
 [<c1549f43>] ? sysenter_past_esp+0x40/0x6a
---[ end trace 6ebc60ef3981792f ]---
```

Such stack traces provide enough information to identify the line inside the Kernel's source code where the bug happened. Depending on the severity of the issue, it may also contain the word **Oops**, as on this one:

```
BUG: unable to handle kernel NULL pointer dereference at   (null)
IP: [<c06969d4>] iret_exc+0x7d0/0xa59
*pdpt = 000000002258a001 *pde = 0000000000000000
Oops: 0002 [#1] PREEMPT SMP
...
```

Despite being an **Oops** or some other sort of stack trace, the offended line is usually required to identify

and handle the bug. Along this chapter, we'll refer to "Oops" for all kinds of stack traces that need to be analized.

Note:

ksymoops is useless on 2.6 or upper. Please use the Oops in its original format (from dmesg, etc). Ignore any references in this or other docs to "decoding the Oops" or "running it through ksymoops". If you post an Oops from 2.6+ that has been run through ksymoops, people will just tell you to repost it.

Where is the Oops message is located?

Normally the Oops text is read from the kernel buffers by klogd and handed to `syslogd` which writes it to a syslog file, typically `/var/log/messages` (depends on `/etc/syslog.conf`). On systems with systemd, it may also be stored by the `journald` daemon, and accessed by running `journalctl` command.

Sometimes `klogd` dies, in which case you can run `dmesg > file` to read the data from the kernel buffers and save it. Or you can `cat /proc/kmsg > file`, however you have to break in to stop the transfer, `kmsg` is a "never ending file".

If the machine has crashed so badly that you cannot enter commands or the disk is not available then you have three options:

1. Hand copy the text from the screen and type it in after the machine has restarted. Messy but it is the only option if you have not planned for a crash. Alternatively, you can take a picture of the screen with a digital camera - not nice, but better than nothing. If the messages scroll off the top of the console, you may find that booting with a higher resolution (eg, vga=791) will allow you to read more of the text. (Caveat: This needs `vesafb`, so won't help for 'early' oopses)
2. Boot with a serial console (see *Documentation/admin-guide/serial-console.rst*), run a null modem to a second machine and capture the output there using your favourite communication program. Minicom works well.
3. Use Kdump (see Documentation/kdump/kdump.txt), extract the kernel ring buffer from old memory with using dmesg gdbmacro in Documentation/kdump/gdbmacros.txt.

Finding the bug's location

Reporting a bug works best if you point the location of the bug at the Kernel source file. There are two methods for doing that. Usually, using gdb is easier, but the Kernel should be pre-compiled with debug info.

gdb

The GNU debug (gdb) is the best way to figure out the exact file and line number of the OOPS from the `vmlinux` file.

The usage of gdb works best on a kernel compiled with `CONFIG_DEBUG_INFO`. This can be set by running:

```
$ ./scripts/config -d COMPILE_TEST -e DEBUG_KERNEL -e DEBUG_INFO
```

On a kernel compiled with `CONFIG_DEBUG_INFO`, you can simply copy the EIP value from the OOPS:

```
EIP:    0060:[<c021e50e>]    Not tainted VLI
```

And use GDB to translate that to human-readable form:

```
$ gdb vmlinux
(gdb) l *0xc021e50e
```

If you don't have CONFIG_DEBUG_INFO enabled, you use the function offset from the OOPS:

```
EIP is at vt_ioctl+0xda8/0x1482
```

And recompile the kernel with CONFIG_DEBUG_INFO enabled:

```
$ ./scripts/config -d COMPILE_TEST -e DEBUG_KERNEL -e DEBUG_INFO
$ make vmlinux
$ gdb vmlinux
(gdb) l *vt_ioctl+0xda8
0x1888 is in vt_ioctl (drivers/tty/vt/vt_ioctl.c:293).
288   {
289           struct vc_data *vc = NULL;
290           int ret = 0;
291
292           console_lock();
293           if (VT_BUSY(vc_num))
294                   ret = -EBUSY;
295           else if (vc_num)
296                   vc = vc_deallocate(vc_num);
297           console_unlock();
```

or, if you want to be more verbose:

```
(gdb) p vt_ioctl
$1 = {int (struct tty_struct *, unsigned int, unsigned long)} 0xae0 <vt_ioctl>
(gdb) l *0xae0+0xda8
```

You could, instead, use the object file:

```
$ make drivers/tty/
$ gdb drivers/tty/vt/vt_ioctl.o
(gdb) l *vt_ioctl+0xda8
```

If you have a call trace, such as:

```
Call Trace:
 [<ffffffff8802c8e9>] :jbd:log_wait_commit+0xa3/0xf5
 [<ffffffff810482d9>] autoremove_wake_function+0x0/0x2e
 [<ffffffff8802770b>] :jbd:journal_stop+0x1be/0x1ee
 ...
```

this shows the problem likely in the :jbd: module. You can load that module in gdb and list the relevant code:

```
$ gdb fs/jbd/jbd.ko
(gdb) l *log_wait_commit+0xa3
```

Note:

You can also do the same for any function call at the stack trace, like this one:

```
[<f80bc9ca>] ? dvb_usb_adapter_frontend_exit+0x3a/0x70 [dvb_usb]
```

The position where the above call happened can be seen with:

```
$ gdb drivers/media/usb/dvb-usb/dvb-usb.o
(gdb) l *dvb_usb_adapter_frontend_exit+0x3a
```

objdump

To debug a kernel, use objdump and look for the hex offset from the crash output to find the valid line of code/assembler. Without debug symbols, you will see the assembler code for the routine shown, but if your kernel has debug symbols the C code will also be available. (Debug symbols can be enabled in the kernel hacking menu of the menu configuration.) For example:

```
$ objdump -r -S -l --disassemble net/dccp/ipv4.o
```

Note:

You need to be at the top level of the kernel tree for this to pick up your C files.

If you don't have access to the code you can also debug on some crash dumps e.g. crash dump output as shown by Dave Miller:

```
EIP is at  +0x14/0x4c0
 ...
Code: 44 24 04 e8 6f 05 00 00 e9 e8 fe ff ff 8d 76 00 8d bc 27 00 00
00 00 55 57  56 53 81 ec bc 00 00 00 8b ac 24 d0 00 00 00 8b 5d 08
<8b> 83 3c 01 00 00 89 44  24 14 8b 45 28 85 c0 89 44 24 18 0f 85

Put the bytes into a "foo.s" file like this:

       .text
       .globl foo
foo:
       .byte  .... /* bytes from Code: part of OOPS dump */

Compile it with "gcc -c -o foo.o foo.s" then look at the output of
"objdump --disassemble foo.o".

Output:

ip_queue_xmit:
    push       %ebp
    push       %edi
    push       %esi
    push       %ebx
    sub        $0xbc, %esp
    mov        0xd0(%esp), %ebp        ! %ebp = arg0 (skb)
    mov        0x8(%ebp), %ebx         ! %ebx = skb->sk
    mov        0x13c(%ebx), %eax       ! %eax = inet_sk(sk)->opt
```

Reporting the bug

Once you find where the bug happened, by inspecting its location, you could either try to fix it yourself or report it upstream.

In order to report it upstream, you should identify the mailing list used for the development of the affected code. This can be done by using the `get_maintainer.pl` script.

For example, if you find a bug at the gspca's conex.c file, you can get their maintainers with:

```
$ ./scripts/get_maintainer.pl -f drivers/media/usb/gspca/sonixj.c
Hans Verkuil <hverkuil@xs4all.nl> (odd fixer:GSPCA USB WEBCAM DRIVER,commit_signer:1/1=100%)
Mauro Carvalho Chehab <mchehab@kernel.org> (maintainer:MEDIA INPUT INFRASTRUCTURE (V4L/DVB),
  ↪commit_signer:1/1=100%)
Tejun Heo <tj@kernel.org> (commit_signer:1/1=100%)
Bhaktipriya Shridhar <bhaktipriya96@gmail.com> (commit_signer:1/1=100%,authored:1/1=100%,added_
  ↪lines:4/4=100%,removed_lines:9/9=100%)
linux-media@vger.kernel.org (open list:GSPCA USB WEBCAM DRIVER)
linux-kernel@vger.kernel.org (open list)
```

Please notice that it will point to:

- The last developers that touched on the source code. On the above example, Tejun and Bhaktipriya (in this specific case, none really envolved on the development of this file);
- The driver maintainer (Hans Verkuil);
- The subsystem maintainer (Mauro Carvalho Chehab)
- The driver and/or subsystem mailing list (linux-media@vger.kernel.org);
- the Linux Kernel mailing list (linux-kernel@vger.kernel.org).

Usually, the fastest way to have your bug fixed is to report it to mailing list used for the development of the code (linux-media ML) copying the driver maintainer (Hans).

If you are totally stumped as to whom to send the report, and `get_maintainer.pl` didn't provide you anything useful, send it to linux-kernel@vger.kernel.org.

Thanks for your help in making Linux as stable as humanly possible.

Fixing the bug

If you know programming, you could help us by not only reporting the bug, but also providing us with a solution. After all open source is about sharing what you do and don't you want to be recognised for your genius?

If you decide to take this way, once you have worked out a fix please submit it upstream.

Please do read ref:*Documentation/process/submitting-patches.rst <submittingpatches>* though to help your code get accepted.

Notes on Oops tracing with `klogd`

In order to help Linus and the other kernel developers there has been substantial support incorporated into `klogd` for processing protection faults. In order to have full support for address resolution at least version 1.3-pl3 of the `sysklogd` package should be used.

When a protection fault occurs the `klogd` daemon automatically translates important addresses in the kernel log messages to their symbolic equivalents. This translated kernel message is then forwarded

through whatever reporting mechanism klogd is using. The protection fault message can be simply cut out of the message files and forwarded to the kernel developers.

Two types of address resolution are performed by klogd. The first is static translation and the second is dynamic translation. Static translation uses the System.map file in much the same manner that ksymoops does. In order to do static translation the klogd daemon must be able to find a system map file at daemon initialization time. See the klogd man page for information on how klogd searches for map files.

Dynamic address translation is important when kernel loadable modules are being used. Since memory for kernel modules is allocated from the kernel's dynamic memory pools there are no fixed locations for either the start of the module or for functions and symbols in the module.

The kernel supports system calls which allow a program to determine which modules are loaded and their location in memory. Using these system calls the klogd daemon builds a symbol table which can be used to debug a protection fault which occurs in a loadable kernel module.

At the very minimum klogd will provide the name of the module which generated the protection fault. There may be additional symbolic information available if the developer of the loadable module chose to export symbol information from the module.

Since the kernel module environment can be dynamic there must be a mechanism for notifying the klogd daemon when a change in module environment occurs. There are command line options available which allow klogd to signal the currently executing daemon that symbol information should be refreshed. See the klogd manual page for more information.

A patch is included with the sysklogd distribution which modifies the modules-2.0.0 package to automatically signal klogd whenever a module is loaded or unloaded. Applying this patch provides essentially seamless support for debugging protection faults which occur with kernel loadable modules.

The following is an example of a protection fault in a loadable module processed by klogd:

```
Aug 29 09:51:01 blizard kernel: Unable to handle kernel paging request at virtual address␣
 ↪f15e97cc
Aug 29 09:51:01 blizard kernel: current->tss.cr3 = 0062d000, %cr3 = 0062d000
Aug 29 09:51:01 blizard kernel: *pde = 00000000
Aug 29 09:51:01 blizard kernel: Oops: 0002
Aug 29 09:51:01 blizard kernel: CPU:    0
Aug 29 09:51:01 blizard kernel: EIP:    0010:[oops:_oops+16/3868]
Aug 29 09:51:01 blizard kernel: EFLAGS: 00010212
Aug 29 09:51:01 blizard kernel: eax: 315e97cc   ebx: 003a6f80   ecx: 001be77b   edx: 00237c0c
Aug 29 09:51:01 blizard kernel: esi: 00000000   edi: bffffdb3   ebp: 00589f90   esp: 00589f8c
Aug 29 09:51:01 blizard kernel: ds: 0018   es: 0018   fs: 002b   gs: 002b   ss: 0018
Aug 29 09:51:01 blizard kernel: Process oops_test (pid: 3374, process nr: 21,␣
 ↪stackpage=00589000)
Aug 29 09:51:01 blizard kernel: Stack: 315e97cc 00589f98 0100b0b4 bffffed4 0012e38e 00240c64␣
 ↪003a6f80 00000001
Aug 29 09:51:01 blizard kernel:        00000000 00237810 bfffff00 0010a7fa 00000003 00000001␣
 ↪00000000 bfffff00
Aug 29 09:51:01 blizard kernel:        bffffdb3 bffffed4 ffffffda 0000002b 0007002b 0000002b␣
 ↪0000002b 00000036
Aug 29 09:51:01 blizard kernel: Call Trace: [oops:_oops_ioctl+48/80] [_sys_ioctl+254/272] [_
 ↪system_call+82/128]
Aug 29 09:51:01 blizard kernel: Code: c7 00 05 00 00 00 eb 08 90 90 90 90 90 90 90 90 89 ec 5d␣
 ↪c3
```

```
Dr. G.W. Wettstein           Oncology Research Div. Computing Facility
Roger Maris Cancer Center    INTERNET: greg@wind.rmcc.com
820 4th St. N.
Fargo, ND  58122
Phone: 701-234-7556
```

CHAPTER

SEVEN

BISECTING A BUG

Last updated: 28 October 2016

Introduction

Always try the latest kernel from kernel.org and build from source. If you are not confident in doing that please report the bug to your distribution vendor instead of to a kernel developer.

Finding bugs is not always easy. Have a go though. If you can't find it don't give up. Report as much as you have found to the relevant maintainer. See MAINTAINERS for who that is for the subsystem you have worked on.

Before you submit a bug report read *Documentation/admin-guide/reporting-bugs.rst* .

Devices not appearing

Often this is caused by udev/systemd. Check that first before blaming it on the kernel.

Finding patch that caused a bug

Using the provided tools with `git` makes finding bugs easy provided the bug is reproducible.

Steps to do it:

- build the Kernel from its git source
- start bisect with [1]:

```
$ git bisect start
```

- mark the broken changeset with:

```
$ git bisect bad [commit]
```

- mark a changeset where the code is known to work with:

```
$ git bisect good [commit]
```

- rebuild the Kernel and test
- interact with git bisect by using either:

[1] You can, optionally, provide both good and bad arguments at git start with `git bisect start [BAD] [GOOD]`

```
$ git bisect good
```

or:

```
$ git bisect bad
```

depending if the bug happened on the changeset you're testing

- After some interactions, git bisect will give you the changeset that likely caused the bug.
- For example, if you know that the current version is bad, and version 4.8 is good, you could do:

```
$ git bisect start
$ git bisect bad                 # Current version is bad
$ git bisect good v4.8
```

For further references, please read:

- The man page for `git-bisect`
- Fighting regressions with git bisect
- Fully automated bisecting with "git bisect run"
- Using Git bisect to figure out when brokenness was introduced

CHAPTER

EIGHT

TAINTED KERNELS

Some oops reports contain the string **'Tainted: '** after the program counter. This indicates that the kernel has been tainted by some mechanism. The string is followed by a series of position-sensitive characters, each representing a particular tainted value.

1. 'G' if all modules loaded have a GPL or compatible license, 'P' if any proprietary module has been loaded. Modules without a MODULE_LICENSE or with a MODULE_LICENSE that is not recognised by insmod as GPL compatible are assumed to be proprietary.
2. F if any module was force loaded by `insmod -f`, ' ' if all modules were loaded normally.
3. S if the oops occurred on an SMP kernel running on hardware that hasn't been certified as safe to run multiprocessor. Currently this occurs only on various Athlons that are not SMP capable.
4. R if a module was force unloaded by `rmmod -f`, ' ' if all modules were unloaded normally.
5. M if any processor has reported a Machine Check Exception, ' ' if no Machine Check Exceptions have occurred.
6. B if a page-release function has found a bad page reference or some unexpected page flags.
7. U if a user or user application specifically requested that the Tainted flag be set, ' ' otherwise.
8. D if the kernel has died recently, i.e. there was an OOPS or BUG.
9. A if the ACPI table has been overridden.
10. W if a warning has previously been issued by the kernel. (Though some warnings may set more specific taint flags.)
11. C if a staging driver has been loaded.
12. I if the kernel is working around a severe bug in the platform firmware (BIOS or similar).
13. O if an externally-built ("out-of-tree") module has been loaded.
14. E if an unsigned module has been loaded in a kernel supporting module signature.
15. L if a soft lockup has previously occurred on the system.
16. K if the kernel has been live patched.

The primary reason for the **'Tainted: '** string is to tell kernel debuggers if this is a clean kernel or if anything unusual has occurred. Tainting is permanent: even if an offending module is unloaded, the tainted value remains to indicate that the kernel is not trustworthy.

CHAPTER

NINE

RAMOOPS OOPS/PANIC LOGGER

Sergiu Iordache <sergiu@chromium.org>

Updated: 17 November 2011

Introduction

Ramoops is an oops/panic logger that writes its logs to RAM before the system crashes. It works by logging oopses and panics in a circular buffer. Ramoops needs a system with persistent RAM so that the content of that area can survive after a restart.

Ramoops concepts

Ramoops uses a predefined memory area to store the dump. The start and size and type of the memory area are set using three variables:

- `mem_address` for the start
- `mem_size` for the size. The memory size will be rounded down to a power of two.
- `mem_type` to specifiy if the memory type (default is pgprot_writecombine).

Typically the default value of `mem_type=0` should be used as that sets the pstore mapping to pgprot_writecombine. Setting `mem_type=1` attempts to use `pgprot_noncached`, which only works on some platforms. This is because pstore depends on atomic operations. At least on ARM, pgprot_noncached causes the memory to be mapped strongly ordered, and atomic operations on strongly ordered memory are implementation defined, and won't work on many ARMs such as omaps.

The memory area is divided into `record_size` chunks (also rounded down to power of two) and each oops/panic writes a `record_size` chunk of information.

Dumping both oopses and panics can be done by setting 1 in the `dump_oops` variable while setting 0 in that variable dumps only the panics.

The module uses a counter to record multiple dumps but the counter gets reset on restart (i.e. new dumps after the restart will overwrite old ones).

Ramoops also supports software ECC protection of persistent memory regions. This might be useful when a hardware reset was used to bring the machine back to life (i.e. a watchdog triggered). In such cases, RAM may be somewhat corrupt, but usually it is restorable.

Setting the parameters

Setting the ramoops parameters can be done in several different manners:

A. Use the module parameters (which have the names of the variables described as before). For quick debugging, you can also reserve parts of memory during boot and then use the reserved memory for ramoops. For example, assuming a machine with > 128 MB of memory, the following kernel command line will tell the kernel to use only the first 128 MB of memory, and place ECC-protected ramoops region at 128 MB boundary:

```
mem=128M ramoops.mem_address=0x8000000 ramoops.ecc=1
```

B. Use Device Tree bindings, as described in `Documentation/devicetree/bindings/reserved-memory/admin-guide/ramoops.rst`. For example:

```
reserved-memory {
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;

        ramoops@8f000000 {
                compatible = "ramoops";
                reg = <0 0x8f000000 0 0x100000>;
                record-size = <0x4000>;
                console-size = <0x4000>;
        };
};
```

C. Use a platform device and set the platform data. The parameters can then be set through that platform data. An example of doing that is:

```
#include <linux/pstore_ram.h>
[...]

static struct ramoops_platform_data ramoops_data = {
      .mem_size               = <...>,
      .mem_address            = <...>,
      .mem_type               = <...>,
      .record_size            = <...>,
      .dump_oops              = <...>,
      .ecc                    = <...>,
};

static struct platform_device ramoops_dev = {
      .name = "ramoops",
      .dev = {
              .platform_data = &ramoops_data,
      },
};

[... inside a function ...]
int ret;

ret = platform_device_register(&ramoops_dev);
if (ret) {
      printk(KERN_ERR "unable to register platform device\n");
      return ret;
}
```

You can specify either RAM memory or peripheral devices' memory. However, when specifying RAM, be sure to reserve the memory by issuing memblock_reserve() very early in the architecture code, e.g.:

```
#include <linux/memblock.h>

memblock_reserve(ramoops_data.mem_address, ramoops_data.mem_size);
```

Dump format

The data dump begins with a header, currently defined as ==== followed by a timestamp and a new line. The dump then continues with the actual data.

Reading the data

The dump data can be read from the pstore filesystem. The format for these files is `dmesg-ramoops-N`, where N is the record number in memory. To delete a stored record from RAM, simply unlink the respective pstore file.

Persistent function tracing

Persistent function tracing might be useful for debugging software or hardware related hangs. The functions call chain log is stored in a `ftrace-ramoops` file. Here is an example of usage:

```
# mount -t debugfs debugfs /sys/kernel/debug/
# echo 1 > /sys/kernel/debug/pstore/record_ftrace
# reboot -f
[...]
# mount -t pstore pstore /mnt/
# tail /mnt/ftrace-ramoops
0 ffffffff8101ea64  ffffffff8101bcda  native_apic_mem_read <- disconnect_bsp_APIC+0x6a/0xc0
0 ffffffff8101ea44  ffffffff8101bcf6  native_apic_mem_write <- disconnect_bsp_APIC+0x86/0xc0
0 ffffffff81020084  ffffffff8101a4b5  hpet_disable <- native_machine_shutdown+0x75/0x90
0 ffffffff81005f94  ffffffff8101a4bb  iommu_shutdown_noop <- native_machine_shutdown+0x7b/0x90
0 ffffffff8101a6a1  ffffffff8101a437  native_machine_emergency_restart <- native_machine_
 ↪restart+0x37/0x40
0 ffffffff811f9876  ffffffff8101a73a  acpi_reboot <- native_machine_emergency_restart+0xaa/0x1e0
0 ffffffff8101a514  ffffffff8101a772  mach_reboot_fixups <- native_machine_emergency_
 ↪restart+0xe2/0x1e0
0 ffffffff811d9c54  ffffffff8101a7a0  __const_udelay <- native_machine_emergency_restart+0x110/
 ↪0x1e0
0 ffffffff811d9c34  ffffffff811d9c80  __delay <- __const_udelay+0x30/0x40
0 ffffffff811d9d14  ffffffff811d9c3f  delay_tsc <- __delay+0xf/0x20
```

CHAPTER

TEN

DYNAMIC DEBUG

Introduction

This document describes how to use the dynamic debug (dyndbg) feature.

Dynamic debug is designed to allow you to dynamically enable/disable kernel code to obtain additional kernel information. Currently, if `CONFIG_DYNAMIC_DEBUG` is set, then all `pr_debug()`/`dev_dbg()` and `print_hex_dump_debug()`/`print_hex_dump_bytes()` calls can be dynamically enabled per-callsite.

If `CONFIG_DYNAMIC_DEBUG` is not set, `print_hex_dump_debug()` is just shortcut for `print_hex_dump(KERN_DEBUG)`.

For `print_hex_dump_debug()`/`print_hex_dump_bytes()`, format string is its `prefix_str` argument, if it is constant string; or hexdump in case `prefix_str` is build dynamically.

Dynamic debug has even more useful features:

- Simple query language allows turning on and off debugging statements by matching any combination of 0 or 1 of:
  - source filename
  - function name
  - line number (including ranges of line numbers)
  - module name
  - format string
- Provides a debugfs control file: `<debugfs>/dynamic_debug/control` which can be read to display the complete list of known debug statements, to help guide you

Controlling dynamic debug Behaviour

The behaviour of `pr_debug()`/`dev_dbg()` are controlled via writing to a control file in the 'debugfs' filesystem. Thus, you must first mount the debugfs filesystem, in order to make use of this feature. Subsequently, we refer to the control file as: `<debugfs>/dynamic_debug/control`. For example, if you want to enable printing from source file `svcsock.c`, line 1603 you simply do:

```
nullarbor:~ # echo 'file svcsock.c line 1603 +p' >
                              <debugfs>/dynamic_debug/control
```

If you make a mistake with the syntax, the write will fail thus:

```
nullarbor:~ # echo 'file svcsock.c wtf 1 +p' >
                              <debugfs>/dynamic_debug/control
-bash: echo: write error: Invalid argument
```

Viewing Dynamic Debug Behaviour

You can view the currently configured behaviour of all the debug statements via:

```
nullarbor:~ # cat <debugfs>/dynamic_debug/control
# filename:lineno [module]function flags format
/usr/src/packages/BUILD/sgi-enhancednfs-1.4/default/net/sunrpc/svc_rdma.c:323 [svcxprt_rdma]svc_
  ↪rdma_cleanup =_ "SVCRDMA Module Removed, deregister RPC RDMA transport\012"
/usr/src/packages/BUILD/sgi-enhancednfs-1.4/default/net/sunrpc/svc_rdma.c:341 [svcxprt_rdma]svc_
  ↪rdma_init =_ "\011max_inline       : %d\012"
/usr/src/packages/BUILD/sgi-enhancednfs-1.4/default/net/sunrpc/svc_rdma.c:340 [svcxprt_rdma]svc_
  ↪rdma_init =_ "\011sq_depth         : %d\012"
/usr/src/packages/BUILD/sgi-enhancednfs-1.4/default/net/sunrpc/svc_rdma.c:338 [svcxprt_rdma]svc_
  ↪rdma_init =_ "\011max_requests     : %d\012"
...
```

You can also apply standard Unix text manipulation filters to this data, e.g.:

```
nullarbor:~ # grep -i rdma <debugfs>/dynamic_debug/control  | wc -l
62

nullarbor:~ # grep -i tcp <debugfs>/dynamic_debug/control | wc -l
42
```

The third column shows the currently enabled flags for each debug statement callsite (see below for definitions of the flags). The default value, with no flags enabled, is =_. So you can view all the debug statement callsites with any non-default flags:

```
nullarbor:~ # awk '$3 != "=_"' <debugfs>/dynamic_debug/control
# filename:lineno [module]function flags format
/usr/src/packages/BUILD/sgi-enhancednfs-1.4/default/net/sunrpc/svcsock.c:1603 [sunrpc]svc_send␣
  ↪p "svc_process: st_sendto returned %d\012"
```

Command Language Reference

At the lexical level, a command comprises a sequence of words separated by spaces or tabs. So these are all equivalent:

```
nullarbor:~ # echo -n 'file svcsock.c line 1603 +p' >
                              <debugfs>/dynamic_debug/control
nullarbor:~ # echo -n '  file   svcsock.c     line  1603 +p  ' >
                              <debugfs>/dynamic_debug/control
nullarbor:~ # echo -n 'file svcsock.c line 1603 +p' >
                              <debugfs>/dynamic_debug/control
```

Command submissions are bounded by a write() system call. Multiple commands can be written together, separated by ; or \n:

```
~# echo "func pnpacpi_get_resources +p; func pnp_assign_mem +p" \
   > <debugfs>/dynamic_debug/control
```

If your query set is big, you can batch them too:

```
~# cat query-batch-file > <debugfs>/dynamic_debug/control
```

A another way is to use wildcard. The match rule support * (matches zero or more characters) and ? (matches exactly one character).For example, you can match all usb drivers:

```
~# echo "file drivers/usb/* +p" > <debugfs>/dynamic_debug/control
```

At the syntactical level, a command comprises a sequence of match specifications, followed by a flags change specification:

```
command ::= match-spec* flags-spec
```

The match-spec's are used to choose a subset of the known pr_debug() callsites to which to apply the flags-spec. Think of them as a query with implicit ANDs between each pair. Note that an empty list of match-specs will select all debug statement callsites.

A match specification comprises a keyword, which controls the attribute of the callsite to be compared, and a value to compare against. Possible keywords are::

```
match-spec ::= 'func' string |
               'file' string |
               'module' string |
               'format' string |
               'line' line-range

line-range ::= lineno |
               '-'lineno |
               lineno'-' |
               lineno'-'lineno

lineno ::= unsigned-int
```

Note:

line-range cannot contain space, e.g. "1-30" is valid range but "1 - 30" is not.

The meanings of each keyword are:

func The given string is compared against the function name of each callsite. Example:

```
func svc_tcp_accept
```

file The given string is compared against either the full pathname, the src-root relative pathname, or the basename of the source file of each callsite. Examples:

```
file svcsock.c
file kernel/freezer.c
file /usr/src/packages/BUILD/sgi-enhancednfs-1.4/default/net/sunrpc/svcsock.c
```

module The given string is compared against the module name of each callsite. The module name is the string as seen in `lsmod`, i.e. without the directory or the `.ko` suffix and with `-` changed to `_`. Examples:

```
module sunrpc
module nfsd
```

format The given string is searched for in the dynamic debug format string. Note that the string does not need to match the entire format, only some part. Whitespace and other special characters can be escaped using C octal character escape `\ooo` notation, e.g. the space character is `\040`. Alternatively, the string can be enclosed in double quote characters (`"`) or single quote characters (`'`). Examples:

```
format svcrdma:         // many of the NFS/RDMA server pr_debugs
format readahead        // some pr_debugs in the readahead cache
format nfsd:\040SETATTR // one way to match a format with whitespace
```

```
format "nfsd: SETATTR"  // a neater way to match a format with whitespace
format 'nfsd: SETATTR'  // yet another way to match a format with whitespace
```

line The given line number or range of line numbers is compared against the line number of each `pr_debug()` callsite. A single line number matches the callsite line number exactly. A range of line numbers matches any callsite between the first and last line number inclusive. An empty first number means the first line in the file, an empty line number means the last number in the file. Examples:

```
line 1603           // exactly line 1603
line 1600-1605      // the six lines from line 1600 to line 1605
line -1605          // the 1605 lines from line 1 to line 1605
line 1600-          // all lines from line 1600 to the end of the file
```

The flags specification comprises a change operation followed by one or more flag characters. The change operation is one of the characters:

```
-    remove the given flags
+    add the given flags
=    set the flags to the given flags
```

The flags are:

```
p    enables the pr_debug() callsite.
f    Include the function name in the printed message
l    Include line number in the printed message
m    Include module name in the printed message
t    Include thread ID in messages not generated from interrupt context
_    No flags are set. (Or'd with others on input)
```

For `print_hex_dump_debug()` and `print_hex_dump_bytes()`, only `p` flag have meaning, other flags ignored.

For display, the flags are preceded by `=` (mnemonic: what the flags are currently equal to).

Note the regexp `^[-+=][flmpt_]+$` matches a flags specification. To clear all flags at once, use `=_` or `-flmpt`.

Debug messages during Boot Process

To activate debug messages for core code and built-in modules during the boot process, even before userspace and debugfs exists, use `dyndbg="QUERY"`, `module.dyndbg="QUERY"`, or `ddebug_query="QUERY"` (`ddebug_query` is obsoleted by dyndbg, and deprecated). QUERY follows the syntax described above, but must not exceed 1023 characters. Your bootloader may impose lower limits.

These dyndbg params are processed just after the ddebug tables are processed, as part of the arch_initcall. Thus you can enable debug messages in all code run after this arch_initcall via this boot parameter.

On an x86 system for example ACPI enablement is a subsys_initcall and:

```
dyndbg="file ec.c +p"
```

will show early Embedded Controller transactions during ACPI setup if your machine (typically a laptop) has an Embedded Controller. PCI (or other devices) initialization also is a hot candidate for using this boot parameter for debugging purposes.

If `foo` module is not built-in, `foo.dyndbg` will still be processed at boot time, without effect, but will be reprocessed when module is loaded later. `dyndbg_query=` and bare `dyndbg=` are only processed at boot.

Debug Messages at Module Initialization Time

When `modprobe foo` is called, modprobe scans `/proc/cmdline` for `foo.params`, strips `foo.`, and passes them to the kernel along with params given in modprobe args or `/etc/modprob.d/*.conf` files, in the following order:

1. parameters given via `/etc/modprobe.d/*.conf`:

```
options foo dyndbg=+pt
options foo dyndbg # defaults to +p
```

2. `foo.dyndbg` as given in boot args, `foo.` is stripped and passed:

```
foo.dyndbg=" func bar +p; func buz +mp"
```

3. args to modprobe:

```
modprobe foo dyndbg==pmf # override previous settings
```

These dyndbg queries are applied in order, with last having final say. This allows boot args to override or modify those from `/etc/modprobe.d` (sensible, since 1 is system wide, 2 is kernel or boot specific), and modprobe args to override both.

In the `foo.dyndbg="QUERY"` form, the query must exclude `module foo`. `foo` is extracted from the param-name, and applied to each query in `QUERY`, and only 1 match-spec of each type is allowed.

The dyndbg option is a “fake” module parameter, which means:

- modules do not need to define it explicitly
- every module gets it tacitly, whether they use pr_debug or not
- it doesn’t appear in `/sys/module/$module/parameters/` To see it, grep the control file, or inspect `/proc/cmdline`.

For `CONFIG_DYNAMIC_DEBUG` kernels, any settings given at boot-time (or enabled by `-DDEBUG` flag during compilation) can be disabled later via the sysfs interface if the debug messages are no longer needed:

```
echo "module module_name -p" > <debugfs>/dynamic_debug/control
```

Examples

```
// enable the message at line 1603 of file svcsock.c
nullarbor:~ # echo -n 'file svcsock.c line 1603 +p' >
                              <debugfs>/dynamic_debug/control

// enable all the messages in file svcsock.c
nullarbor:~ # echo -n 'file svcsock.c +p' >
                              <debugfs>/dynamic_debug/control

// enable all the messages in the NFS server module
nullarbor:~ # echo -n 'module nfsd +p' >
                              <debugfs>/dynamic_debug/control

// enable all 12 messages in the function svc_process()
nullarbor:~ # echo -n 'func svc_process +p' >
                              <debugfs>/dynamic_debug/control

// disable all 12 messages in the function svc_process()
nullarbor:~ # echo -n 'func svc_process -p' >
                              <debugfs>/dynamic_debug/control
```

```
// enable messages for NFS calls READ, READLINK, READDIR and READDIR+.
nullarbor:~ # echo -n 'format "nfsd: READ" +p' >
                              <debugfs>/dynamic_debug/control

// enable messages in files of which the paths include string "usb"
nullarbor:~ # echo -n '*usb* +p' > <debugfs>/dynamic_debug/control

// enable all messages
nullarbor:~ # echo -n '+p' > <debugfs>/dynamic_debug/control

// add module, function to all enabled messages
nullarbor:~ # echo -n '+mf' > <debugfs>/dynamic_debug/control

// boot-args example, with newlines and comments for readability
Kernel command line: ...
  // see whats going on in dyndbg=value processing
  dynamic_debug.verbose=1
  // enable pr_debugs in 2 builtins, #cmt is stripped
  dyndbg="module params +p #cmt ; module sys +p"
  // enable pr_debugs in 2 functions in a module loaded later
  pc87360.dyndbg="func pc87360_init_device +p; func pc87360_find +p"
```

CHAPTER

ELEVEN

EXPLAINING THE DREADED “NO INIT FOUND.” BOOT HANG MESSAGE

OK, so you’ve got this pretty unintuitive message (currently located in init/main.c) and are wondering what the H*** went wrong. Some high-level reasons for failure (listed roughly in order of execution) to load the init binary are:

1. Unable to mount root FS
2. init binary doesn’t exist on rootfs
3. broken console device
4. binary exists but dependencies not available
5. binary cannot be loaded

Detailed explanations:

1. Set “debug” kernel parameter (in bootloader config file or CONFIG_CMDLINE) to get more detailed kernel messages.
2. make sure you have the correct root FS type (and `root=` kernel parameter points to the correct partition), required drivers such as storage hardware (such as SCSI or USB!) and filesystem (ext3, jffs2 etc.) are builtin (alternatively as modules, to be pre-loaded by an initrd)
3. Possibly a conflict in `console= setup` –> initial console unavailable. E.g. some serial consoles are unreliable due to serial IRQ issues (e.g. missing interrupt-based configuration). Try using a different `console= device` or e.g. `netconsole=`.
4. e.g. required library dependencies of the init binary such as `/lib/ld-linux.so.2` missing or broken. Use `readelf -d <INIT>|grep NEEDED` to find out which libraries are required.
5. make sure the binary’s architecture matches your hardware. E.g. i386 vs. x86_64 mismatch, or trying to load x86 on ARM hardware. In case you tried loading a non-binary file here (shell script?), you should make sure that the script specifies an interpreter in its shebang header line (`#!/...`) that is fully working (including its library dependencies). And before tackling scripts, better first test a simple non-script binary such as `/bin/sh` and confirm its successful execution. To find out more, add code `to init/main.c` to display kernel_execve()s return values.

Please extend this explanation whenever you find new failure causes (after all loading the init binary is a CRITICAL and hard transition step which needs to be made as painless as possible), then submit patch to LKML. Further TODOs:

- Implement the various `run_init_process()` invocations via a struct array which can then store the `kernel_execve()` result value and on failure log it all by iterating over **all** results (very important usability fix).
- try to make the implementation itself more helpful in general, e.g. by providing additional error messages at affected places.

Andreas Mohr <andi at lisas period de>

This is the beginning of a section with information of interest to application developers. Documents covering various aspects of the kernel ABI will be found here.

CHAPTER

TWELVE

RULES ON HOW TO ACCESS INFORMATION IN SYSFS

The kernel-exported sysfs exports internal kernel implementation details and depends on internal kernel structures and layout. It is agreed upon by the kernel developers that the Linux kernel does not provide a stable internal API. Therefore, there are aspects of the sysfs interface that may not be stable across kernel releases.

To minimize the risk of breaking users of sysfs, which are in most cases low-level userspace applications, with a new kernel release, the users of sysfs must follow some rules to use an as-abstract-as-possible way to access this filesystem. The current udev and HAL programs already implement this and users are encouraged to plug, if possible, into the abstractions these programs provide instead of accessing sysfs directly.

But if you really do want or need to access sysfs directly, please follow the following rules and then your programs should work with future versions of the sysfs interface.

- **Do not use libsysfs** It makes assumptions about sysfs which are not true. Its API does not offer any abstraction, it exposes all the kernel driver-core implementation details in its own API. Therefore it is not better than reading directories and opening the files yourself. Also, it is not actively maintained, in the sense of reflecting the current kernel development. The goal of providing a stable interface to sysfs has failed; it causes more problems than it solves. It violates many of the rules in this document.
- **sysfs is always at `/sys`** Parsing `/proc/mounts` is a waste of time. Other mount points are a system configuration bug you should not try to solve. For test cases, possibly support a SYSFS_PATH environment variable to overwrite the application's behavior, but never try to search for sysfs. Never try to mount it, if you are not an early boot script.
- **devices are only "devices"** There is no such thing like class-, bus-, physical devices, interfaces, and such that you can rely on in userspace. Everything is just simply a "device". Class-, bus-, physical, ... types are just kernel implementation details which should not be expected by applications that look for devices in sysfs.

  The properties of a device are:

  - devpath (`/devices/pci0000:00/0000:00:1d.1/usb2/2-2/2-2:1.0`)
    * identical to the DEVPATH value in the event sent from the kernel at device creation and removal
    * the unique key to the device at that point in time
    * the kernel's path to the device directory without the leading `/sys`, and always starting with a slash
    * all elements of a devpath must be real directories. Symlinks pointing to /sys/devices must always be resolved to their real target and the target path must be used to access the device. That way the devpath to the device matches the devpath of the kernel used at event time.
    * using or exposing symlink values as elements in a devpath string is a bug in the application
  - kernel name (`sda`, `tty`, `0000:00:1f.2`, ...)

    * a directory name, identical to the last element of the devpath
    * applications need to handle spaces and characters like ! in the name
  - subsystem (block, tty, pci, ...)
    * simple string, never a path or a link
    * retrieved by reading the "subsystem"-link and using only the last element of the target path
  - driver (tg3, ata_piix, uhci_hcd)
    * a simple string, which may contain spaces, never a path or a link
    * it is retrieved by reading the "driver"-link and using only the last element of the target path
    * devices which do not have "driver"-link just do not have a driver; copying the driver value in a child device context is a bug in the application
  - attributes
    * the files in the device directory or files below subdirectories of the same device directory
    * accessing attributes reached by a symlink pointing to another device, like the "device"-link, is a bug in the application

  Everything else is just a kernel driver-core implementation detail that should not be assumed to be stable across kernel releases.

- **Properties of parent devices never belong into a child device.** Always look at the parent devices themselves for determining device context properties. If the device eth0 or sda does not have a "driver"-link, then this device does not have a driver. Its value is empty. Never copy any property of the parent-device into a child-device. Parent device properties may change dynamically without any notice to the child device.
- **Hierarchy in a single device tree** There is only one valid place in sysfs where hierarchy can be examined and this is below: /sys/devices. It is planned that all device directories will end up in the tree below this directory.
- **Classification by subsystem** There are currently three places for classification of devices: /sys/block, /sys/class and /sys/bus. It is planned that these will not contain any device directories themselves, but only flat lists of symlinks pointing to the unified /sys/devices tree. All three places have completely different rules on how to access device information. It is planned to merge all three classification directories into one place at /sys/subsystem, following the layout of the bus directories. All buses and classes, including the converted block subsystem, will show up there. The devices belonging to a subsystem will create a symlink in the "devices" directory at /sys/subsystem/<name>/devices,

  If /sys/subsystem exists, /sys/bus, /sys/class and /sys/block can be ignored. If it does not exist, you always have to scan all three places, as the kernel is free to move a subsystem from one place to the other, as long as the devices are still reachable by the same subsystem name.

  Assuming /sys/class/<subsystem> and /sys/bus/<subsystem>, or /sys/block and /sys/class/block are not interchangeable is a bug in the application.
- **Block** The converted block subsystem at /sys/class/block or /sys/subsystem/block will contain the links for disks and partitions at the same level, never in a hierarchy. Assuming the block subsystem to contain only disks and not partition devices in the same flat list is a bug in the application.
- **"device"-link and <subsystem>:<kernel name>-links** Never depend on the "device"-link. The "device"-link is a workaround for the old layout, where class devices are not created in /sys/devices/ like the bus devices. If the link-resolving of a device directory does not end in /sys/devices/, you can use the "device"-link to find the parent devices in /sys/devices/,

That is the single valid use of the “device”-link; it must never appear in any path as an element. Assuming the existence of the “device”-link for a device in `/sys/devices/` is a bug in the application. Accessing `/sys/class/net/eth0/device` is a bug in the application.

Never depend on the class-specific links back to the `/sys/class` directory. These links are also a workaround for the design mistake that class devices are not created in `/sys/devices`. If a device directory does not contain directories for child devices, these links may be used to find the child devices in `/sys/class`. That is the single valid use of these links; they must never appear in any path as an element. Assuming the existence of these links for devices which are real child device directories in the `/sys/devices` tree is a bug in the application.

It is planned to remove all these links when all class device directories live in `/sys/devices`.

- **Position of devices along device chain can change.** Never depend on a specific parent device position in the devpath, or the chain of parent devices. The kernel is free to insert devices into the chain. You must always request the parent device you are looking for by its subsystem value. You need to walk up the chain until you find the device that matches the expected subsystem. Depending on a specific position of a parent device or exposing relative paths using `../` to access the chain of parents is a bug in the application.
- **When reading and writing sysfs device attribute files, avoid dependency** on specific error codes wherever possible. This minimizes coupling to the error handling implementation within the kernel.

  In general, failures to read or write sysfs device attributes shall propagate errors wherever possible. Common errors include, but are not limited to:

  `-EIO`: The read or store operation is not supported, typically returned by the sysfs system itself if the read or store pointer is `NULL`.

  `-ENXIO`: The read or store operation failed

  Error codes will not be changed without good reason, and should a change to error codes result in user-space breakage, it will be fixed, or the the offending change will be reverted.

  Userspace applications can, however, expect the format and contents of the attribute files to remain consistent in the absence of a version attribute change in the context of a given attribute.

The rest of this manual consists of various unordered guides on how to configure specific aspects of kernel behavior to your liking.

CHAPTER

THIRTEEN

USING THE INITIAL RAM DISK (INITRD)

Written 1996,2000 by Werner Almesberger <werner.almesberger@epfl.ch> and Hans Lermen <lermen@fgan.de>

initrd provides the capability to load a RAM disk by the boot loader. This RAM disk can then be mounted as the root file system and programs can be run from it. Afterwards, a new root file system can be mounted from a different device. The previous root (from initrd) is then moved to a directory and can be subsequently unmounted.

initrd is mainly designed to allow system startup to occur in two phases, where the kernel comes up with a minimum set of compiled-in drivers, and where additional modules are loaded from initrd.

This document gives a brief overview of the use of initrd. A more detailed discussion of the boot process can be found in [1].

Operation

When using initrd, the system typically boots as follows:

1. the boot loader loads the kernel and the initial RAM disk
2. the kernel converts initrd into a “normal” RAM disk and frees the memory used by initrd
3. if the root device is not `/dev/ram0`, the old (deprecated) change_root procedure is followed. see the “Obsolete root change mechanism” section below.
4. root device is mounted. if it is `/dev/ram0`, the initrd image is then mounted as root
5. /sbin/init is executed (this can be any valid executable, including shell scripts; it is run with uid 0 and can do basically everything init can do).
6. init mounts the “real” root file system
7. init places the root file system at the root directory using the pivot_root system call
8. init execs the `/sbin/init` on the new root filesystem, performing the usual boot sequence
9. the initrd file system is removed

Note that changing the root directory does not involve unmounting it. It is therefore possible to leave processes running on initrd during that procedure. Also note that file systems mounted under initrd continue to be accessible.

Boot command-line options

initrd adds the following new options:

[1] Almesberger, Werner; “Booting Linux: The History and the Future” http://www.almesberger.net/cv/papers/ols2k-9.ps.gz

```
initrd=<path>    (e.g. LOADLIN)

  Loads the specified file as the initial RAM disk. When using LILO, you
  have to specify the RAM disk image file in /etc/lilo.conf, using the
  INITRD configuration variable.

noinitrd

  initrd data is preserved but it is not converted to a RAM disk and
  the "normal" root file system is mounted. initrd data can be read
  from /dev/initrd. Note that the data in initrd can have any structure
  in this case and doesn't necessarily have to be a file system image.
  This option is used mainly for debugging.

  Note: /dev/initrd is read-only and it can only be used once. As soon
  as the last process has closed it, all data is freed and /dev/initrd
  can't be opened anymore.

root=/dev/ram0

  initrd is mounted as root, and the normal boot procedure is followed,
  with the RAM disk mounted as root.
```

Compressed cpio images

Recent kernels have support for populating a ramdisk from a compressed cpio archive. On such systems, the creation of a ramdisk image doesn't need to involve special block devices or loopbacks; you merely create a directory on disk with the desired initrd content, cd to that directory, and run (as an example):

```
find . | cpio --quiet -H newc -o | gzip -9 -n > /boot/imagefile.img
```

Examining the contents of an existing image file is just as simple:

```
mkdir /tmp/imagefile
cd /tmp/imagefile
gzip -cd /boot/imagefile.img | cpio -imd --quiet
```

Installation

First, a directory for the initrd file system has to be created on the "normal" root file system, e.g.:

```
# mkdir /initrd
```

The name is not relevant. More details can be found on the *pivot_root(2)* man page.

If the root file system is created during the boot procedure (i.e. if you're building an install floppy), the root file system creation procedure should create the `/initrd` directory.

If initrd will not be mounted in some cases, its content is still accessible if the following device has been created:

```
# mknod /dev/initrd b 1 250
# chmod 400 /dev/initrd
```

Second, the kernel has to be compiled with RAM disk support and with support for the initial RAM disk enabled. Also, at least all components needed to execute programs from initrd (e.g. executable format and file system) must be compiled into the kernel.

Third, you have to create the RAM disk image. This is done by creating a file system on a block device, copying files to it as needed, and then copying the content of the block device to the initrd file. With recent kernels, at least three types of devices are suitable for that:

- a floppy disk (works everywhere but it's painfully slow)
- a RAM disk (fast, but allocates physical memory)
- a loopback device (the most elegant solution)

We'll describe the loopback device method:

1. make sure loopback block devices are configured into the kernel
2. create an empty file system of the appropriate size, e.g.:

```
# dd if=/dev/zero of=initrd bs=300k count=1
# mke2fs -F -m0 initrd
```

   (if space is critical, you may want to use the Minix FS instead of Ext2)

3. mount the file system, e.g.:

```
# mount -t ext2 -o loop initrd /mnt
```

4. create the console device:

```
# mkdir /mnt/dev
# mknod /mnt/dev/console c 5 1
```

5. copy all the files that are needed to properly use the initrd environment. Don't forget the most important file, `/sbin/init`

   Note:

   `/sbin/init` permissions must include "x" (execute).

6. correct operation the initrd environment can frequently be tested even without rebooting with the command:

```
# chroot /mnt /sbin/init
```

   This is of course limited to initrds that do not interfere with the general system state (e.g. by reconfiguring network interfaces, overwriting mounted devices, trying to start already running demons, etc. Note however that it is usually possible to use pivot_root in such a chroot'ed initrd environment.)

7. unmount the file system:

```
# umount /mnt
```

8. the initrd is now in the file "initrd". Optionally, it can now be compressed:

```
# gzip -9 initrd
```

For experimenting with initrd, you may want to take a rescue floppy and only add a symbolic link from `/sbin/init` to `/bin/sh`. Alternatively, you can try the experimental newlib environment [2] to create a small initrd.

Finally, you have to boot the kernel and load initrd. Almost all Linux boot loaders support initrd. Since the boot process is still compatible with an older mechanism, the following boot command line parameters have to be given:

[2] newlib package (experimental), with initrd example https://www.sourceware.org/newlib/

```
root=/dev/ram0 rw
```

(rw is only necessary if writing to the initrd file system.)

With LOADLIN, you simply execute:

```
LOADLIN <kernel> initrd=<disk_image>
```

e.g.:

```
LOADLIN C:\LINUX\BZIMAGE initrd=C:\LINUX\INITRD.GZ root=/dev/ram0 rw
```

With LILO, you add the option INITRD=<path> to either the global section or to the section of the respective kernel in /etc/lilo.conf, and pass the options using APPEND, e.g.:

```
image = /bzImage
  initrd = /boot/initrd.gz
  append = "root=/dev/ram0 rw"
```

and run /sbin/lilo

For other boot loaders, please refer to the respective documentation.

Now you can boot and enjoy using initrd.

Changing the root device

When finished with its duties, init typically changes the root device and proceeds with starting the Linux system on the "real" root device.

The procedure involves the following steps:

- mounting the new root file system
- turning it into the root file system
- removing all accesses to the old (initrd) root file system
- unmounting the initrd file system and de-allocating the RAM disk

Mounting the new root file system is easy: it just needs to be mounted on a directory under the current root. Example:

```
# mkdir /new-root
# mount -o ro /dev/hda1 /new-root
```

The root change is accomplished with the pivot_root system call, which is also available via the pivot_root utility (see *pivot_root(8)* man page; pivot_root is distributed with util-linux version 2.10h or higher [3]). pivot_root moves the current root to a directory under the new root, and puts the new root at its place. The directory for the old root must exist before calling pivot_root. Example:

```
# cd /new-root
# mkdir initrd
# pivot_root . initrd
```

Now, the init process may still access the old root via its executable, shared libraries, standard input/output/error, and its current root directory. All these references are dropped by the following command:

```
# exec chroot . what-follows <dev/console >dev/console 2>&1
```

[3] util-linux: Miscellaneous utilities for Linux https://www.kernel.org/pub/linux/utils/util-linux/

Where what-follows is a program under the new root, e.g. `/sbin/init` If the new root file system will be used with udev and has no valid `/dev` directory, udev must be initialized before invoking chroot in order to provide `/dev/console`.

Note: implementation details of pivot_root may change with time. In order to ensure compatibility, the following points should be observed:

- before calling pivot_root, the current directory of the invoking process should point to the new root directory
- use . as the first argument, and the _relative_ path of the directory for the old root as the second argument
- a chroot program must be available under the old and the new root
- chroot to the new root afterwards
- use relative paths for dev/console in the exec command

Now, the initrd can be unmounted and the memory allocated by the RAM disk can be freed:

```
# umount /initrd
# blockdev --flushbufs /dev/ram0
```

It is also possible to use initrd with an NFS-mounted root, see the *pivot_root(8)* man page for details.

Usage scenarios

The main motivation for implementing initrd was to allow for modular kernel configuration at system installation. The procedure would work as follows:

1. system boots from floppy or other media with a minimal kernel (e.g. support for RAM disks, initrd, a.out, and the Ext2 FS) and loads initrd
2. `/sbin/init` determines what is needed to (1) mount the “real” root FS (i.e. device type, device drivers, file system) and (2) the distribution media (e.g. CD-ROM, network, tape, ...). This can be done by asking the user, by auto-probing, or by using a hybrid approach.
3. `/sbin/init` loads the necessary kernel modules
4. `/sbin/init` creates and populates the root file system (this doesn’t have to be a very usable system yet)
5. `/sbin/init` invokes `pivot_root` to change the root file system and execs - via chroot - a program that continues the installation
6. the boot loader is installed
7. the boot loader is configured to load an initrd with the set of modules that was used to bring up the system (e.g. `/initrd` can be modified, then unmounted, and finally, the image is written from `/dev/ram0` or `/dev/rd/0` to a file)
8. now the system is bootable and additional installation tasks can be performed

The key role of initrd here is to re-use the configuration data during normal system operation without requiring the use of a bloated “generic” kernel or re-compiling or re-linking the kernel.

A second scenario is for installations where Linux runs on systems with different hardware configurations in a single administrative domain. In such cases, it is desirable to generate only a small set of kernels (ideally only one) and to keep the system-specific part of configuration information as small as possible. In this case, a common initrd could be generated with all the necessary modules. Then, only `/sbin/init` or a file read by it would have to be different.

A third scenario is more convenient recovery disks, because information like the location of the root FS partition doesn’t have to be provided at boot time, but the system loaded from initrd can invoke a user-friendly dialog and it can also perform some sanity checks (or even some form of auto-detection).

Last not least, CD-ROM distributors may use it for better installation from CD, e.g. by using a boot floppy and bootstrapping a bigger RAM disk via initrd from CD; or by booting via a loader like LOADLIN or directly from the CD-ROM, and loading the RAM disk from CD without need of floppies.

Obsolete root change mechanism

The following mechanism was used before the introduction of pivot_root. Current kernels still support it, but you should _not_ rely on its continued availability.

It works by mounting the "real" root device (i.e. the one set with rdev in the kernel image or with root=... at the boot command line) as the root file system when linuxrc exits. The initrd file system is then unmounted, or, if it is still busy, moved to a directory `/initrd`, if such a directory exists on the new root file system.

In order to use this mechanism, you do not have to specify the boot command options root, init, or rw. (If specified, they will affect the real root file system, not the initrd environment.)

If /proc is mounted, the "real" root device can be changed from within linuxrc by writing the number of the new root FS device to the special file /proc/sys/kernel/real-root-dev, e.g.:

```
# echo 0x301 >/proc/sys/kernel/real-root-dev
```

Note that the mechanism is incompatible with NFS and similar file systems.

This old, deprecated mechanism is commonly called `change_root`, while the new, supported mechanism is called `pivot_root`.

Mixed change_root and pivot_root mechanism

In case you did not want to use `root=/dev/ram0` to trigger the pivot_root mechanism, you may create both `/linuxrc` and `/sbin/init` in your initrd image.

`/linuxrc` would contain only the following:

```
#! /bin/sh
mount -n -t proc proc /proc
echo 0x0100 >/proc/sys/kernel/real-root-dev
umount -n /proc
```

Once linuxrc exited, the kernel would mount again your initrd as root, this time executing `/sbin/init`. Again, it would be the duty of this init to build the right environment (maybe using the `root=` device passed on the cmdline) before the final execution of the real `/sbin/init`.

Resources

CHAPTER

FOURTEEN

LINUX SERIAL CONSOLE

To use a serial port as console you need to compile the support into your kernel - by default it is not compiled in. For PC style serial ports it's the config option next to menu option:

Character devices → Serial drivers → 8250/16550 and compatible serial support → Console on 8250/16550 and compatible serial port

You must compile serial support into the kernel and not as a module.

It is possible to specify multiple devices for console output. You can define a new kernel command line option to select which device(s) to use for console output.

The format of this option is:

```
console=device,options

device:         tty0 for the foreground virtual console
                ttyX for any other virtual console
                ttySx for a serial port
                lp0 for the first parallel port
                ttyUSB0 for the first USB serial device

options:        depend on the driver. For the serial port this
                defines the baudrate/parity/bits/flow control of
                the port, in the format BBBBPNF, where BBBB is the
                speed, P is parity (n/o/e), N is number of bits,
                and F is flow control ('r' for RTS). Default is
                9600n8. The maximum baudrate is 115200.
```

You can specify multiple console= options on the kernel command line. Output will appear on all of them. The last device will be used when you open `/dev/console`. So, for example:

```
console=ttyS1,9600 console=tty0
```

defines that opening `/dev/console` will get you the current foreground virtual console, and kernel messages will appear on both the VGA console and the 2nd serial port (ttyS1 or COM2) at 9600 baud.

Note that you can only define one console per device type (serial, video).

If no console device is specified, the first device found capable of acting as a system console will be used. At this time, the system first looks for a VGA card and then for a serial port. So if you don't have a VGA card in your system the first serial port will automatically become the console.

You will need to create a new device to use `/dev/console`. The official `/dev/console` is now character device 5,1.

(You can also use a network device as a console. See `Documentation/networking/netconsole.txt` for information on that.)

Here's an example that will use `/dev/ttyS1` (COM2) as the console. Replace the sample values as needed.

1. Create `/dev/console` (real console) and `/dev/tty0` (master virtual console):

```
cd /dev
rm -f console tty0
mknod -m 622 console c 5 1
mknod -m 622 tty0 c 4 0
```

2. LILO can also take input from a serial device. This is a very useful option. To tell LILO to use the serial port: In lilo.conf (global section):

```
serial  = 1,9600n8 (ttyS1, 9600 bd, no parity, 8 bits)
```

3. Adjust to kernel flags for the new kernel, again in lilo.conf (kernel section):

```
append = "console=ttyS1,9600"
```

4. Make sure a getty runs on the serial port so that you can login to it once the system is done booting. This is done by adding a line like this to `/etc/inittab` (exact syntax depends on your getty):

```
S1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
```

5. Init and `/etc/ioctl.save`

   Sysvinit remembers its stty settings in a file in `/etc`, called `/etc/ioctl.save`. REMOVE THIS FILE before using the serial console for the first time, because otherwise init will probably set the baudrate to 38400 (baudrate of the virtual console).

6. `/dev/console` and X Programs that want to do something with the virtual console usually open `/dev/console`. If you have created the new `/dev/console` device, and your console is NOT the virtual console some programs will fail. Those are programs that want to access the VT interface, and use `/dev/console instead of /dev/tty0`. Some of those programs are:

```
Xfree86, svgalib, gpm, SVGATextMode
```

   It should be fixed in modern versions of these programs though.

   Note that if you boot without a `console=` option (or with `console=/dev/tty0`), `/dev/console` is the same as `/dev/tty0`. In that case everything will still work.

7. Thanks

   Thanks to Geert Uytterhoeven <geert@linux-m68k.org> for porting the patches from 2.1.4x to 2.1.6x for taking care of the integration of these patches into m68k, ppc and alpha.

Miquel van Smoorenburg <miquels@cistron.nl>, 11-Jun-2000

CHAPTER

FIFTEEN

LINUX BRAILLE CONSOLE

To get early boot messages on a braille device (before userspace screen readers can start), you first need to compile the support for the usual serial console (see *Documentation/admin-guide/serial-console.rst*), and for braille device (in *Device Drivers* → *Accessibility support* → *Console on braille device*).

Then you need to specify a `console=brl`, option on the kernel command line, the format is:

```
console=brl,serial_options...
```

where `serial_options...` are the same as described in *Documentation/admin-guide/serial-console.rst* .

So for instance you can use `console=brl,ttyS0` if the braille device is connected to the first serial port, and `console=brl,ttyS0,115200` to override the baud rate to 115200, etc.

By default, the braille device will just show the last kernel message (console mode). To review previous messages, press the Insert key to switch to the VT review mode. In review mode, the arrow keys permit to browse in the VT content, `PAGE-UP`/`PAGE-DOWN` keys go at the top/bottom of the screen, and the `HOME` key goes back to the cursor, hence providing very basic screen reviewing facility.

Sound feedback can be obtained by adding the `braille_console.sound=1` kernel parameter.

For simplicity, only one braille console can be enabled, other uses of `console=brl,...` will be discarded. Also note that it does not interfere with the console selection mechanism described in *Documentation/admin-guide/serial-console.rst* .

For now, only the VisioBraille device is supported.

Samuel Thibault <samuel.thibault@ens-lyon.org>

CHAPTER

SIXTEEN

PARPORT

The parport code provides parallel-port support under Linux. This includes the ability to share one port between multiple device drivers.

You can pass parameters to the parport code to override its automatic detection of your hardware. This is particularly useful if you want to use IRQs, since in general these can't be autoprobed successfully. By default IRQs are not used even if they **can** be probed. This is because there are a lot of people using the same IRQ for their parallel port and a sound card or network card.

The parport code is split into two parts: generic (which deals with port-sharing) and architecture-dependent (which deals with actually using the port).

Parport as modules

If you load the *parport*' code as a module, say:

```
# insmod parport
```

to load the generic parport code. You then must load the architecture-dependent code with (for example):

```
# insmod parport_pc io=0x3bc,0x378,0x278 irq=none,7,auto
```

to tell the parport code that you want three PC-style ports, one at 0x3bc with no IRQ, one at 0x378 using IRQ 7, and one at 0x278 with an auto-detected IRQ. Currently, PC-style (parport_pc), Sun bpp, Amiga, Atari, and MFC3 hardware is supported.

PCI parallel I/O card support comes from parport_pc. Base I/O addresses should not be specified for supported PCI cards since they are automatically detected.

modprobe

If you use modprobe , you will find it useful to add lines as below to a configuration file in /etc/modprobe.d/ directory:

```
alias parport_lowlevel parport_pc
options parport_pc io=0x378,0x278 irq=7,auto
```

modprobe will load parport_pc (with the options io=0x378,0x278 irq=7,auto) whenever a parallel port device driver (such as lp) is loaded.

Note that these are example lines only! You shouldn't in general need to specify any options to parport_pc in order to be able to use a parallel port.

Parport probe [optional]

In 2.2 kernels there was a module called `parport_probe`, which was used for collecting IEEE 1284 device ID information. This has now been enhanced and now lives with the IEEE 1284 support. When a parallel port is detected, the devices that are connected to it are analysed, and information is logged like this:

```
parport0: Printer, BJC-210 (Canon)
```

The probe information is available from files in `/proc/sys/dev/parport/`.

Parport linked into the kernel statically

If you compile the `parport` code into the kernel, then you can use kernel boot parameters to get the same effect. Add something like the following to your LILO command line:

```
parport=0x3bc parport=0x378,7 parport=0x278,auto,nofifo
```

You can have many `parport=...` statements, one for each port you want to add. Adding `parport=0` to the kernel command-line will disable parport support entirely. Adding `parport=auto` to the kernel command-line will make `parport` use any IRQ lines or DMA channels that it auto-detects.

Files in /proc

If you have configured the `/proc` filesystem into your kernel, you will see a new directory entry: `/proc/sys/dev/parport`. In there will be a directory entry for each parallel port for which parport is configured. In each of those directories are a collection of files describing that parallel port.

The `/proc/sys/dev/parport` directory tree looks like:

```
parport
|-- default
|   |-- spintime
|   `-- timeslice
|-- parport0
|   |-- autoprobe
|   |-- autoprobe0
|   |-- autoprobe1
|   |-- autoprobe2
|   |-- autoprobe3
|   |-- devices
|   |   |-- active
|   |   `-- lp
|   |       `-- timeslice
|   |-- base-addr
|   |-- irq
|   |-- dma
|   |-- modes
|   `-- spintime
`-- parport1
|-- autoprobe
|-- autoprobe0
|-- autoprobe1
|-- autoprobe2
|-- autoprobe3
|-- devices
|   |-- active
|   `-- ppa
|       `-- timeslice
```

```
|-- base-addr
|-- irq
|-- dma
|-- modes
`-- spintime
```

File	Contents
devices/active	A list of the device drivers using that port. A "+" will appear by the name of the device currently using the port (it might not appear against any). The string "none" means that there are no device drivers using that port.
base-addr	Parallel port's base address, or addresses if the port has more than one in which case they are separated with tabs. These values might not have any sensible meaning for some ports.
irq	Parallel port's IRQ, or -1 if none is being used.
dma	Parallel port's DMA channel, or -1 if none is being used.
modes	Parallel port's hardware modes, comma-separated, meaning: • **PCSPP** PC-style SPP registers are available. • **TRISTATE** Port is bidirectional. • **COMPAT** Hardware acceleration for printers is available and will be used. • **EPP** Hardware acceleration for EPP protocol is available and will be used. • **ECP** Hardware acceleration for ECP protocol is available and will be used. • **DMA** DMA is available and will be used. Note that the current implementation will only take advantage of COMPAT and ECP modes if it has an IRQ line to use.
autoprobe	Any IEEE-1284 device ID information that has been acquired from the (non-IEEE 1284.3) device.
autoprobe[0-3]	IEEE 1284 device ID information retrieved from daisy-chain devices that conform to IEEE 1284.3.
spintime	The number of microseconds to busy-loop while waiting for the peripheral to respond. You might find that adjusting this improves performance, depending on your peripherals. This is a port-wide setting, i.e. it applies to all devices on a particular port.
timeslice	The number of milliseconds that a device driver is allowed to keep a port claimed for. This is advisory, and driver can ignore it if it must.
default/*	The defaults for spintime and timeslice. When a new port is registered, it picks up the default spintime. When a new device is registered, it picks up the default timeslice.

Device drivers

Once the parport code is initialised, you can attach device drivers to specific ports. Normally this happens automatically; if the lp driver is loaded it will create one lp device for each port found. You can override this, though, by using parameters either when you load the lp driver:

```
# insmod lp parport=0,2
```

or on the LILO command line:

```
lp=parport0 lp=parport2
```

Both the above examples would inform lp that you want /dev/lp0 to be the first parallel port, and /dev/lp1 to be the **third** parallel port, with no lp device associated with the second port (parport1). Note that this is different to the way older kernels worked; there used to be a static association between the I/O port address and the device name, so /dev/lp0 was always the port at 0x3bc. This is no longer the case - if you only have one port, it will default to being /dev/lp0, regardless of base address.

Also:

- If you selected the IEEE 1284 support at compile time, you can say `lp=auto` on the kernel command line, and lp will create devices only for those ports that seem to have printers attached.
- If you give PLIP the `timid` parameter, either with `plip=timid` on the command line, or with `insmod plip timid=1` when using modules, it will avoid any ports that seem to be in use by other devices.
- IRQ autoprobing works only for a few port types at the moment.

Reporting printer problems with parport

If you are having problems printing, please go through these steps to try to narrow down where the problem area is.

When reporting problems with parport, really you need to give all of the messages that `parport_pc` spits out when it initialises. There are several code paths:

- polling
- interrupt-driven, protocol in software
- interrupt-driven, protocol in hardware using PIO
- interrupt-driven, protocol in hardware using DMA

The kernel messages that `parport_pc` logs give an indication of which code path is being used. (They could be a lot better actually..)

For normal printer protocol, having IEEE 1284 modes enabled or not should not make a difference.

To turn off the 'protocol in hardware' code paths, disable `CONFIG_PARPORT_PC_FIFO`. Note that when they are enabled they are not necessarily **used**; it depends on whether the hardware is available, enabled by the BIOS, and detected by the driver.

So, to start with, disable `CONFIG_PARPORT_PC_FIFO`, and load `parport_pc` with `irq=none`. See if printing works then. It really should, because this is the simplest code path.

If that works fine, try with `io=0x378 irq=7` (adjust for your hardware), to make it use interrupt-driven in-software protocol.

If **that** works fine, then one of the hardware modes isn't working right. Enable `CONFIG_FIFO` (no, it isn't a module option, and yes, it should be), set the port to ECP mode in the BIOS and note the DMA channel, and try with:

```
io=0x378 irq=7 dma=none (for PIO)
io=0x378 irq=7 dma=3 (for DMA)
```

---

philb@gnu.org tim@cyberelk.net

CHAPTER

SEVENTEEN

RAID ARRAYS

Boot time assembly of RAID arrays

Tools that manage md devices can be found at http://www.kernel.org/pub/linux/utils/raid/

You can boot with your md device with the following kernel command lines:

for old raid arrays without persistent superblocks:

```
md=<md device no.>,<raid level>,<chunk size factor>,<fault level>,dev0,dev1,...,devn
```

for raid arrays with persistent superblocks:

```
md=<md device no.>,dev0,dev1,...,devn
```

or, to assemble a partitionable array:

```
md=d<md device no.>,dev0,dev1,...,devn
```

md device no.

The number of the md device

md device no.	device
0	md0
1	md1
2	md2
3	md3
4	md4

raid level

level of the RAID array

raid level	level
-1	linear mode
0	striped mode

other modes are only supported with persistent super blocks

chunk size factor

(raid-0 and raid-1 only)

Set the chunk size as 4k << n.

fault level

Totally ignored

dev0 to devn

e.g. /dev/hda1, /dev/hdc1, /dev/sda1, /dev/sdb1

A possible loadlin line (Harald Hoyer <HarryH@Royal.Net>) looks like this:

```
e:\loadlin\loadlin e:\zimage root=/dev/md0 md=0,0,4,0,/dev/hdb2,/dev/hdc3 ro
```

Boot time autodetection of RAID arrays

When md is compiled into the kernel (not as module), partitions of type 0xfd are scanned and automatically assembled into RAID arrays. This autodetection may be suppressed with the kernel parameter `raid=noautodetect`. As of kernel 2.6.9, only drives with a type 0 superblock can be autodetected and run at boot time.

The kernel parameter `raid=partitionable` (or `raid=part`) means that all auto-detected arrays are assembled as partitionable.

Boot time assembly of degraded/dirty arrays

If a raid5 or raid6 array is both dirty and degraded, it could have undetectable data corruption. This is because the fact that it is `dirty` means that the parity cannot be trusted, and the fact that it is degraded means that some datablocks are missing and cannot reliably be reconstructed (due to no parity).

For this reason, md will normally refuse to start such an array. This requires the sysadmin to take action to explicitly start the array despite possible corruption. This is normally done with:

```
mdadm --assemble --force ....
```

This option is not really available if the array has the root filesystem on it. In order to support this booting from such an array, md supports a module parameter `start_dirty_degraded` which, when set to 1, bypassed the checks and will allows dirty degraded arrays to be started.

So, to boot with a root filesystem of a dirty degraded raid 5 or 6, use:

```
md-mod.start_dirty_degraded=1
```

Superblock formats

The md driver can support a variety of different superblock formats. Currently, it supports superblock formats `0.90.0` and the `md-1` format introduced in the 2.5 development series.

The kernel will autodetect which format superblock is being used.

Superblock format `0` is treated differently to others for legacy reasons - it is the original superblock format.

General Rules - apply for all superblock formats

An array is `created` by writing appropriate superblocks to all devices.

It is `assembled` by associating each of these devices with an particular md virtual device. Once it is completely assembled, it can be accessed.

An array should be created by a user-space tool. This will write superblocks to all devices. It will usually mark the array as `unclean`, or with some devices missing so that the kernel md driver can create appropriate redundancy (copying in raid 1, parity calculation in raid 4/5).

When an array is assembled, it is first initialized with the SET_ARRAY_INFO ioctl. This contains, in particular, a major and minor version number. The major version number selects which superblock format is to be used. The minor number might be used to tune handling of the format, such as suggesting where on each device to look for the superblock.

Then each device is added using the ADD_NEW_DISK ioctl. This provides, in particular, a major and minor number identifying the device to add.

The array is started with the RUN_ARRAY ioctl.

Once started, new devices can be added. They should have an appropriate superblock written to them, and then be passed in with ADD_NEW_DISK.

Devices that have failed or are not yet active can be detached from an array using HOT_REMOVE_DISK.

Specific Rules that apply to format-0 super block arrays, and arrays with no superblock (non-persistent)

An array can be `created` by describing the array (level, chunksize etc) in a SET_ARRAY_INFO ioctl. This must have `major_version==0` and `raid_disks != 0`.

Then uninitialized devices can be added with ADD_NEW_DISK. The structure passed to ADD_NEW_DISK must specify the state of the device and its role in the array.

Once started with RUN_ARRAY, uninitialized spares can be added with HOT_ADD_DISK.

MD devices in sysfs

md devices appear in sysfs (`/sys`) as regular block devices, e.g.:

```
/sys/block/md0
```

Each `md` device will contain a subdirectory called `md` which contains further md-specific information about the device.

All md devices contain:

level a text file indicating the `raid level`. e.g. raid0, raid1, raid5, linear, multipath, faulty. If no raid level has been set yet (array is still being assembled), the value will reflect whatever has been written to it, which may be a name like the above, or may be a number such as `0`, `5`, etc.

raid_disks a text file with a simple number indicating the number of devices in a fully functional array. If this is not yet known, the file will be empty. If an array is being resized this will contain the new number of devices. Some raid levels allow this value to be set while the array is active. This will reconfigure the array. Otherwise it can only be set while assembling an array. A change to this attribute will not be permitted if it would reduce the size of the array. To reduce the number of drives in an e.g. raid5, the array size must first be reduced by setting the `array_size` attribute.

chunk_size This is the size in bytes for chunks and is only relevant to raid levels that involve striping (0,4,5,6,10). The address space of the array is conceptually divided into chunks and consecutive chunks are striped onto neighbouring devices. The size should be at least PAGE_SIZE (4k) and should be a power of 2. This can only be set while assembling an array

layout The `layout` for the array for the particular level. This is simply a number that is interpretted differently by different levels. It can be written while assembling an array.

array_size This can be used to artificially constrain the available space in the array to be less than is actually available on the combined devices. Writing a number (in Kilobytes) which is less than the available size will set the size. Any reconfiguration of the array (e.g. adding devices) will not cause the size to change. Writing the word `default` will cause the effective size of the array to be whatever size is actually available based on `level`, `chunk_size` and `component_size`.

This can be used to reduce the size of the array before reducing the number of devices in a raid4/5/6, or to support external metadata formats which mandate such clipping.

reshape_position This is either `none` or a sector number within the devices of the array where `reshape` is up to. If this is set, the three attributes mentioned above (raid_disks, chunk_size, layout) can potentially have 2 values, an old and a new value. If these values differ, reading the attribute returns:

```
new (old)
```

and writing will effect the `new` value, leaving the `old` unchanged.

component_size For arrays with data redundancy (i.e. not raid0, linear, faulty, multipath), all components must be the same size - or at least there must a size that they all provide space for. This is a key part or the geometry of the array. It is measured in sectors and can be read from here. Writing to this value may resize the array if the personality supports it (raid1, raid5, raid6), and if the component drives are large enough.

metadata_version This indicates the format that is being used to record metadata about the array. It can be 0.90 (traditional format), 1.0, 1.1, 1.2 (newer format in varying locations) or `none` indicating that the kernel isn't managing metadata at all. Alternately it can be `external:` followed by a string which is set by user-space. This indicates that metadata is managed by a user-space program. Any device failure or other event that requires a metadata update will cause array activity to be suspended until the event is acknowledged.

resync_start The point at which resync should start. If no resync is needed, this will be a very large number (or `none` since 2.6.30-rc1). At array creation it will default to 0, though starting the array as `clean` will set it much larger.

new_dev This file can be written but not read. The value written should be a block device number as major:minor. e.g. 8:0 This will cause that device to be attached to the array, if it is available. It will then appear at md/dev-XXX (depending on the name of the device) and further configuration is then possible.

safe_mode_delay When an md array has seen no write requests for a certain period of time, it will be marked as `clean`. When another write request arrives, the array is marked as `dirty` before the write commences. This is known as `safe_mode`. The `certain period` is controlled by this file which stores the period as a number of seconds. The default is 200msec (0.200). Writing a value of 0 disables safemode.

array_state This file contains a single word which describes the current state of the array. In many cases, the state can be set by writing the word for the desired state, however some states cannot be explicitly set, and some transitions are not allowed.

Select/poll works on this file. All changes except between Active_idle and active (which can be frequent and are not very interesting) are notified. active->active_idle is reported if the metadata is externally managed.

clear No devices, no size, no level

Writing is equivalent to STOP_ARRAY ioctl

inactive May have some settings, but array is not active all IO results in error

When written, doesn't tear down array, but just stops it

suspended (not supported yet) All IO requests will block. The array can be reconfigured.

Writing this, if accepted, will block until array is quiessent

readonly no resync can happen. no superblocks get written.

Write requests fail

read-auto like readonly, but behaves like `clean` on a write request.

clean no pending writes, but otherwise active.

When written to inactive array, starts without resync

If a write request arrives then if metadata is known, mark `dirty` and switch to `active`. if not known, block and switch to write-pending

If written to an active array that has pending writes, then fails.

active fully active: IO and resync can be happening. When written to inactive array, starts with resync

write-pending clean, but writes are blocked waiting for `active` to be written.

active-idle like active, but no writes have been seen for a while (safe_mode_delay).

bitmap/location This indicates where the write-intent bitmap for the array is stored.

It can be one of `none`, `file` or `[+-]N`. `file` may later be extended to `file:/file/name` `[+-]N` means that many sectors from the start of the metadata.

This is replicated on all devices. For arrays with externally managed metadata, the offset is from the beginning of the device.

bitmap/chunksize The size, in bytes, of the chunk which will be represented by a single bit. For RAID456, it is a portion of an individual device. For RAID10, it is a portion of the array. For RAID1, it is both (they come to the same thing).

bitmap/time_base The time, in seconds, between looking for bits in the bitmap to be cleared. In the current implementation, a bit will be cleared between 2 and 3 times `time_base` after all the covered blocks are known to be in-sync.

bitmap/backlog When write-mostly devices are active in a RAID1, write requests to those devices proceed in the background - the filesystem (or other user of the device) does not have to wait for them. `backlog` sets a limit on the number of concurrent background writes. If there are more than this, new writes will by synchronous.

bitmap/metadata This can be either `internal` or `external`.

`internal` is the default and means the metadata for the bitmap is stored in the first 256 bytes of the allocated space and is managed by the md module.

`external` means that bitmap metadata is managed externally to the kernel (i.e. by some userspace program)

bitmap/can_clear This is either `true` or `false`. If `true`, then bits in the bitmap will be cleared when the corresponding blocks are thought to be in-sync. If `false`, bits will never be cleared. This is automatically set to `false` if a write happens on a degraded array, or if the array becomes degraded during a write. When metadata is managed externally, it should be set to true once the array becomes non-degraded, and this fact has been recorded in the metadata.

consistency_policy This indicates how the array maintains consistency in case of unexpected shutdown. It can be:

none Array has no redundancy information, e.g. raid0, linear.

resync Full resync is performed and all redundancy is regenerated when the array is started after unclean shutdown.

bitmap Resync assisted by a write-intent bitmap.

journal For raid4/5/6, journal device is used to log transactions and replay after unclean shutdown.

ppl For raid5 only, Partial Parity Log is used to close the write hole and eliminate resync.

The accepted values when writing to this file are `ppl` and `resync`, used to enable and disable PPL.

As component devices are added to an md array, they appear in the `md` directory as new directories named:

```
dev-XXX
```

where XXX is a name that the kernel knows for the device, e.g. hdb1. Each directory contains:

block a symlink to the block device in /sys/block, e.g.:

```
/sys/block/md0/md/dev-hdb1/block -> ../../../../block/hdb/hdb1
```

super A file containing an image of the superblock read from, or written to, that device.

state A file recording the current state of the device in the array which can be a comma separated list of:

faulty device has been kicked from active use due to a detected fault, or it has unacknowledged bad blocks

in_sync device is a fully in-sync member of the array

writemostly device will only be subject to read requests if there are no other options.

This applies only to raid1 arrays.

blocked device has failed, and the failure hasn't been acknowledged yet by the metadata handler.

Writes that would write to this device if it were not faulty are blocked.

spare device is working, but not a full member.

This includes spares that are in the process of being recovered to

write_error device has ever seen a write error.

want_replacement device is (mostly) working but probably should be replaced, either due to errors or due to user request.

replacement device is a replacement for another active device with same raid_disk.

This list may grow in future.

This can be written to.

Writing `faulty` simulates a failure on the device.

Writing `remove` removes the device from the array.

Writing `writemostly` sets the writemostly flag.

Writing `-writemostly` clears the writemostly flag.

Writing `blocked` sets the `blocked` flag.

Writing -blocked clears the blocked flags and allows writes to complete and possibly simulates an error.

Writing in_sync sets the in_sync flag.

Writing write_error sets writeerrorseen flag.

Writing -write_error clears writeerrorseen flag.

Writing want_replacement is allowed at any time except to a replacement device or a spare. It sets the flag.

Writing -want_replacement is allowed at any time. It clears the flag.

Writing replacement or -replacement is only allowed before starting the array. It sets or clears the flag.

This file responds to select/poll. Any change to faulty or blocked causes an event.

errors An approximate count of read errors that have been detected on this device but have not caused the device to be evicted from the array (either because they were corrected or because they happened while the array was read-only). When using version-1 metadata, this value persists across restarts of the array.

This value can be written while assembling an array thus providing an ongoing count for arrays with metadata managed by userspace.

slot This gives the role that the device has in the array. It will either be none if the device is not active in the array (i.e. is a spare or has failed) or an integer less than the raid_disks number for the array indicating which position it currently fills. This can only be set while assembling an array. A device for which this is set is assumed to be working.

offset This gives the location in the device (in sectors from the start) where data from the array will be stored. Any part of the device before this offset is not touched, unless it is used for storing metadata (Formats 1.1 and 1.2).

size The amount of the device, after the offset, that can be used for storage of data. This will normally be the same as the component_size. This can be written while assembling an array. If a value less than the current component_size is written, it will be rejected.

recovery_start When the device is not in_sync, this records the number of sectors from the start of the device which are known to be correct. This is normally zero, but during a recovery operation it will steadily increase, and if the recovery is interrupted, restoring this value can cause recovery to avoid repeating the earlier blocks. With v1.x metadata, this value is saved and restored automatically.

This can be set whenever the device is not an active member of the array, either before the array is activated, or before the slot is set.

Setting this to none is equivalent to setting in_sync. Setting to any other value also clears the in_sync flag.

bad_blocks This gives the list of all known bad blocks in the form of start address and length (in sectors respectively). If output is too big to fit in a page, it will be truncated. Writing sector length to this file adds new acknowledged (i.e. recorded to disk safely) bad blocks.

unacknowledged_bad_blocks This gives the list of known-but-not-yet-saved-to-disk bad blocks in the same form of bad_blocks. If output is too big to fit in a page, it will be truncated. Writing to this file adds bad blocks without acknowledging them. This is largely for testing.

ppl_sector, ppl_size Location and size (in sectors) of the space used for Partial Parity Log on this device.

An active md device will also contain an entry for each active device in the array. These are named:

```
rdNN
```

where NN is the position in the array, starting from 0. So for a 3 drive array there will be rd0, rd1, rd2. These are symbolic links to the appropriate dev-XXX entry. Thus, for example:

```
cat /sys/block/md*/md/rd*/state
```

will show in_sync on every line.

Active md devices for levels that support data redundancy (1,4,5,6,10) also have

sync_action a text file that can be used to monitor and control the rebuild process. It contains one word which can be one of:

resync redundancy is being recalculated after unclean shutdown or creation

recover a hot spare is being built to replace a failed/missing device

idle nothing is happening

check A full check of redundancy was requested and is happening. This reads all blocks and checks them. A repair may also happen for some raid levels.

repair A full check and repair is happening. This is similar to resync, but was requested by the user, and the write-intent bitmap is NOT used to optimise the process.

This file is writable, and each of the strings that could be read are meaningful for writing.

idle will stop an active resync/recovery etc. There is no guarantee that another resync/recovery may not be automatically started again, though some event will be needed to trigger this.

resync or recovery can be used to restart the corresponding operation if it was stopped with idle.

check and repair will start the appropriate process providing the current state is idle.

This file responds to select/poll. Any important change in the value triggers a poll event. Sometimes the value will briefly be recover if a recovery seems to be needed, but cannot be achieved. In that case, the transition to recover isn't notified, but the transition away is.

degraded This contains a count of the number of devices by which the arrays is degraded. So an optimal array will show 0. A single failed/missing drive will show 1, etc.

This file responds to select/poll, any increase or decrease in the count of missing devices will trigger an event.

mismatch_count When performing check and repair, and possibly when performing resync, md will count the number of errors that are found. The count in mismatch_cnt is the number of sectors that were re-written, or (for check) would have been re-written. As most raid levels work in units of pages rather than sectors, this may be larger than the number of actual errors by a factor of the number of sectors in a page.

bitmap_set_bits If the array has a write-intent bitmap, then writing to this attribute can set bits in the bitmap, indicating that a resync would need to check the corresponding blocks. Either individual numbers or start-end pairs can be written. Multiple numbers can be separated by a space.

Note that the numbers are bit numbers, not block numbers. They should be scaled by the bitmap_chunksize.

sync_speed_min, sync_speed_max This are similar to /proc/sys/dev/raid/speed_limit_{min,max} however they only apply to the particular array.

If no value has been written to these, or if the word `system` is written, then the system-wide value is used. If a value, in kibibytes-per-second is written, then it is used.

When the files are read, they show the currently active value followed by `(local)` or `(system)` depending on whether it is a locally set or system-wide value.

sync_completed This shows the number of sectors that have been completed of whatever the current sync_action is, followed by the number of sectors in total that could need to be processed. The two numbers are separated by a / thus effectively showing one value, a fraction of the process that is complete.

A `select` on this attribute will return when resync completes, when it reaches the current sync_max (below) and possibly at other times.

sync_speed This shows the current actual speed, in K/sec, of the current sync_action. It is averaged over the last 30 seconds.

suspend_lo, suspend_hi The two values, given as numbers of sectors, indicate a range within the array where IO will be blocked. This is currently only supported for raid4/5/6.

sync_min, sync_max The two values, given as numbers of sectors, indicate a range within the array where `check/repair` will operate. Must be a multiple of chunk_size. When it reaches `sync_max` it will pause, rather than complete. You can use `select` or `poll` on `sync_completed` to wait for that number to reach sync_max. Then you can either increase `sync_max`, or can write `idle` to `sync_action`.

The value of `max` for `sync_max` effectively disables the limit. When a resync is active, the value can only ever be increased, never decreased. The value of `0` is the minimum for `sync_min`.

Each active md device may also have attributes specific to the personality module that manages it. These are specific to the implementation of the module and could change substantially if the implementation changes.

These currently include:

stripe_cache_size (currently raid5 only) number of entries in the stripe cache. This is writable, but there are upper and lower limits (32768, 17). Default is 256.

strip_cache_active (currently raid5 only) number of active entries in the stripe cache

preread_bypass_threshold (currently raid5 only) number of times a stripe requiring preread will be bypassed by a stripe that does not require preread. For fairness defaults to 1. Setting this to 0 disables bypass accounting and requires preread stripes to wait until all full-width stripe- writes are complete. Valid values are 0 to stripe_cache_size.

journal_mode (currently raid5 only) The cache mode for raid5. raid5 could include an extra disk for caching. The mode can be “write-throuth” and “write-back”. The default is “write-through”.

CHAPTER

EIGHTEEN

KERNEL MODULE SIGNING FACILITY

Overview

The kernel module signing facility cryptographically signs modules during installation and then checks the signature upon loading the module. This allows increased kernel security by disallowing the loading of unsigned modules or modules signed with an invalid key. Module signing increases security by making it harder to load a malicious module into the kernel. The module signature checking is done by the kernel so that it is not necessary to have trusted userspace bits.

This facility uses X.509 ITU-T standard certificates to encode the public keys involved. The signatures are not themselves encoded in any industrial standard type. The facility currently only supports the RSA public key encryption standard (though it is pluggable and permits others to be used). The possible hash algorithms that can be used are SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 (the algorithm is selected by data in the signature).

Configuring module signing

The module signing facility is enabled by going to the *Enable Loadable Module Support* section of the kernel configuration and turning on:

```
CONFIG_MODULE_SIG       "Module signature verification"
```

This has a number of options available:

1. *Require modules to be validly signed* (`CONFIG_MODULE_SIG_FORCE`)

   This specifies how the kernel should deal with a module that has a signature for which the key is not known or a module that is unsigned.

   If this is off (ie. “permissive”), then modules for which the key is not available and modules that are unsigned are permitted, but the kernel will be marked as being tainted, and the concerned modules will be marked as tainted, shown with the character ‘E’.

   If this is on (ie. “restrictive”), only modules that have a valid signature that can be verified by a public key in the kernel’s possession will be loaded. All other modules will generate an error.

   Irrespective of the setting here, if the module has a signature block that cannot be parsed, it will be rejected out of hand.

2. *Automatically sign all modules* (`CONFIG_MODULE_SIG_ALL`)

   If this is on then modules will be automatically signed during the modules_install phase of a build. If this is off, then the modules must be signed manually using:

   ```
   scripts/sign-file
   ```

3. *Which hash algorithm should modules be signed with?*

   This presents a choice of which hash algorithm the installation phase will sign the modules with:

CONFIG_MODULE_SIG_SHA1	*Sign modules with SHA-1*
CONFIG_MODULE_SIG_SHA224	*Sign modules with SHA-224*
CONFIG_MODULE_SIG_SHA256	*Sign modules with SHA-256*
CONFIG_MODULE_SIG_SHA384	*Sign modules with SHA-384*
CONFIG_MODULE_SIG_SHA512	*Sign modules with SHA-512*

The algorithm selected here will also be built into the kernel (rather than being a module) so that modules signed with that algorithm can have their signatures checked without causing a dependency loop.

4. *File name or PKCS#11 URI of module signing key* (CONFIG_MODULE_SIG_KEY)

   Setting this option to something other than its default of certs/signing_key.pem will disable the autogeneration of signing keys and allow the kernel modules to be signed with a key of your choosing. The string provided should identify a file containing both a private key and its corresponding X.509 certificate in PEM form, or — on systems where the OpenSSL ENGINE_pkcs11 is functional — a PKCS#11 URI as defined by RFC7512. In the latter case, the PKCS#11 URI should reference both a certificate and a private key.

   If the PEM file containing the private key is encrypted, or if the PKCS#11 token requries a PIN, this can be provided at build time by means of the KBUILD_SIGN_PIN variable.

5. *Additional X.509 keys for default system keyring* (CONFIG_SYSTEM_TRUSTED_KEYS)

   This option can be set to the filename of a PEM-encoded file containing additional certificates which will be included in the system keyring by default.

Note that enabling module signing adds a dependency on the OpenSSL devel packages to the kernel build processes for the tool that does the signing.

Generating signing keys

Cryptographic keypairs are required to generate and check signatures. A private key is used to generate a signature and the corresponding public key is used to check it. The private key is only needed during the build, after which it can be deleted or stored securely. The public key gets built into the kernel so that it can be used to check the signatures as the modules are loaded.

Under normal conditions, when CONFIG_MODULE_SIG_KEY is unchanged from its default, the kernel build will automatically generate a new keypair using openssl if one does not exist in the file:

```
certs/signing_key.pem
```

during the building of vmlinux (the public part of the key needs to be built into vmlinux) using parameters in the:

```
certs/x509.genkey
```

file (which is also generated if it does not already exist).

It is strongly recommended that you provide your own x509.genkey file.

Most notably, in the x509.genkey file, the req_distinguished_name section should be altered from the default:

```
[ req_distinguished_name ]
#O = Unspecified company
CN = Build time autogenerated kernel key
#emailAddress = unspecified.user@unspecified.company
```

The generated RSA key size can also be set with:

```
[ req ]
default_bits = 4096
```

It is also possible to manually generate the key private/public files using the x509.genkey key generation configuration file in the root node of the Linux kernel sources tree and the openssl command. The following is an example to generate the public/private key files:

```
openssl req -new -nodes -utf8 -sha256 -days 36500 -batch -x509 \
   -config x509.genkey -outform PEM -out kernel_key.pem \
   -keyout kernel_key.pem
```

The full pathname for the resulting kernel_key.pem file can then be specified in the CONFIG_MODULE_SIG_KEY option, and the certificate and key therein will be used instead of an autogenerated keypair.

Public keys in the kernel

The kernel contains a ring of public keys that can be viewed by root. They're in a keyring called ".system_keyring" that can be seen by:

```
[root@deneb ~]# cat /proc/keys
...
223c7853 I------     1 perm 1f030000     0     0 keyring   .system_keyring: 1
302d2d52 I------     1 perm 1f010000     0     0 asymmetri Fedora kernel signing key:␣
 ↪d69a84e6bce3d216b979e9505b3e3ef9a7118079: X509.RSA a7118079 []
...
```

Beyond the public key generated specifically for module signing, additional trusted certificates can be provided in a PEM-encoded file referenced by the CONFIG_SYSTEM_TRUSTED_KEYS configuration option.

Further, the architecture code may take public keys from a hardware store and add those in also (e.g. from the UEFI key database).

Finally, it is possible to add additional public keys by doing:

```
keyctl padd asymmetric "" [.system_keyring-ID] <[key-file]
```

e.g.:

```
keyctl padd asymmetric "" 0x223c7853 <my_public_key.x509
```

Note, however, that the kernel will only permit keys to be added to .system_keyring _if_ the new key's X.509 wrapper is validly signed by a key that is already resident in the .system_keyring at the time the key was added.

Manually signing modules

To manually sign a module, use the scripts/sign-file tool available in the Linux kernel source tree. The script requires 4 arguments:

1. The hash algorithm (e.g., sha256)
2. The private key filename or PKCS#11 URI
3. The public key filename
4. The kernel module to be signed

The following is an example to sign a kernel module:

```
scripts/sign-file sha512 kernel-signkey.priv \
        kernel-signkey.x509 module.ko
```

The hash algorithm used does not have to match the one configured, but if it doesn't, you should make sure that hash algorithm is either built into the kernel or can be loaded without requiring itself.

If the private key requires a passphrase or PIN, it can be provided in the $KBUILD_SIGN_PIN environment variable.

Signed modules and stripping

A signed module has a digital signature simply appended at the end. The string `~Module signature appended~.` at the end of the module's file confirms that a signature is present but it does not confirm that the signature is valid!

Signed modules are BRITTLE as the signature is outside of the defined ELF container. Thus they MAY NOT be stripped once the signature is computed and attached. Note the entire module is the signed payload, including any and all debug information present at the time of signing.

Loading signed modules

Modules are loaded with insmod, modprobe, `init_module()` or `finit_module()`, exactly as for unsigned modules as no processing is done in userspace. The signature checking is all done within the kernel.

Non-valid signatures and unsigned modules

If `CONFIG_MODULE_SIG_FORCE` is enabled or module.sig_enforce=1 is supplied on the kernel command line, the kernel will only load validly signed modules for which it has a public key. Otherwise, it will also load modules that are unsigned. Any module for which the kernel has a key, but which proves to have a signature mismatch will not be permitted to load.

Any module that has an unparseable signature will be rejected.

Administering/protecting the private key

Since the private key is used to sign modules, viruses and malware could use the private key to sign modules and compromise the operating system. The private key must be either destroyed or moved to a secure location and not kept in the root node of the kernel source tree.

If you use the same private key to sign modules for multiple kernel configurations, you must ensure that the module version information is sufficient to prevent loading a module into a different kernel. Either set `CONFIG_MODVERSIONS=y` or ensure that each configuration has a different kernel release string by changing `EXTRAVERSION` or `CONFIG_LOCALVERSION`.

CHAPTER

NINETEEN

LINUX MAGIC SYSTEM REQUEST KEY HACKS

Documentation for sysrq.c

What is the magic SysRq key?

It is a ‘magical’ key combo you can hit which the kernel will respond to regardless of whatever else it is doing, unless it is completely locked up.

How do I enable the magic SysRq key?

You need to say “yes” to ‘Magic SysRq key (CONFIG_MAGIC_SYSRQ)’ when configuring the kernel. When running a kernel with SysRq compiled in, /proc/sys/kernel/sysrq controls the functions allowed to be invoked via the SysRq key. The default value in this file is set by the CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE config symbol, which itself defaults to 1. Here is the list of possible values in /proc/sys/kernel/sysrq:

- 0 - disable sysrq completely
- 1 - enable all functions of sysrq
- >1 - bitmask of allowed sysrq functions (see below for detailed function description):

```
  2 =   0x2 - enable control of console logging level
  4 =   0x4 - enable control of keyboard (SAK, unraw)
  8 =   0x8 - enable debugging dumps of processes etc.
 16 =  0x10 - enable sync command
 32 =  0x20 - enable remount read-only
 64 =  0x40 - enable signalling of processes (term, kill, oom-kill)
128 =  0x80 - allow reboot/poweroff
256 = 0x100 - allow nicing of all RT tasks
```

You can set the value in the file by the following command:

```
echo "number" >/proc/sys/kernel/sysrq
```

The number may be written here either as decimal or as hexadecimal with the 0x prefix. CONFIG_MAGIC_SYSRQ_DEFAULT_ENABLE must always be written in hexadecimal.

Note that the value of `/proc/sys/kernel/sysrq` influences only the invocation via a keyboard. Invocation of any operation via `/proc/sysrq-trigger` is always allowed (by a user with admin privileges).

How do I use the magic SysRq key?

On x86 - You press the key combo `ALT-SysRq-<command key>`.

Note:

Some keyboards may not have a key labeled 'SysRq'. The 'SysRq' key is also known as the 'Print Screen' key. Also some keyboards cannot handle so many keys being pressed at the same time, so you might have better luck with press `Alt`, press SysRq, release SysRq, press <command key>, release everything.

On SPARC - You press `ALT-STOP-<command key>`, I believe.

On the serial console (PC style standard serial ports only) You send a BREAK, then within 5 seconds a command key. Sending BREAK twice is interpreted as a normal BREAK.

On PowerPC Press `ALT -Print Screen` (or F13) - `<command key>`, `Print Screen` (or F13) - `<command key>` may suffice.

On other If you know of the key combos for other architectures, please let me know so I can add them to this section.

On all write a character to /proc/sysrq-trigger. e.g.:

```
echo t > /proc/sysrq-trigger
```

What are the 'command' keys?

Com-mand	Function
b	Will immediately reboot the system without syncing or unmounting your disks.
c	Will perform a system crash by a NULL pointer dereference. A crashdump will be taken if configured.
d	Shows all locks that are held.
e	Send a SIGTERM to all processes, except for init.
f	Will call the oom killer to kill a memory hog process, but do not panic if nothing can be killed.
g	Used by kgdb (kernel debugger)
h	Will display help (actually any other key than those listed here will display help. but h is easy to remember :-)
i	Send a SIGKILL to all processes, except for init.
j	Forcibly "Just thaw it" - filesystems frozen by the FIFREEZE ioctl.
k	Secure Access Key (SAK) Kills all programs on the current virtual console. NOTE: See important comments below in SAK section.
l	Shows a stack backtrace for all active CPUs.
m	Will dump current memory info to your console.
n	Used to make RT tasks nice-able
o	Will shut your system off (if configured and supported).
p	Will dump the current registers and flags to your console.
q	Will dump per CPU lists of all armed hrtimers (but NOT regular timer_list timers) and detailed information about all clockevent devices.
r	Turns off keyboard raw mode and sets it to XLATE.
s	Will attempt to sync all mounted filesystems.
t	Will dump a list of current tasks and their information to your console.
u	Will attempt to remount all mounted filesystems read-only.
v	Forcefully restores framebuffer console
v	Causes ETM buffer dump [ARM-specific]
w	Dumps tasks that are in uninterruptable (blocked) state.
x	Used by xmon interface on ppc/powerpc platforms. Show global PMU Registers on sparc64. Dump all TLB entries on MIPS.
y	Show global CPU Registers [SPARC-64 specific]
z	Dump the ftrace buffer
0-9	Sets the console log level, controlling which kernel messages will be printed to your console. (0, for example would make it so that only emergency messages like PANICs or OOPSes would make it to your console.)

Okay, so what can I use them for?

Well, unraw(r) is very handy when your X server or a svgalib program crashes.

sak(k) (Secure Access Key) is useful when you want to be sure there is no trojan program running at console which could grab your password when you would try to login. It will kill all programs on given console, thus letting you make sure that the login prompt you see is actually the one from init, not some trojan program.

Important:

In its true form it is not a true SAK like the one in a c2 compliant system, and it should not be mistaken as such.

It seems others find it useful as (System Attention Key) which is useful when you want to exit a program

that will not let you switch consoles. (For example, X or a svgalib program.)

`reboot(b)` is good when you're unable to shut down. But you should also `sync(s)` and `umount(u)` first.

`crash(c)` can be used to manually trigger a crashdump when the system is hung. Note that this just triggers a crash if there is no dump mechanism available.

`sync(s)` is great when your system is locked up, it allows you to sync your disks and will certainly lessen the chance of data loss and fscking. Note that the sync hasn't taken place until you see the "OK" and "Done" appear on the screen. (If the kernel is really in strife, you may not ever get the OK or Done message...)

`umount(u)` is basically useful in the same ways as `sync(s)`. I generally `sync(s)`, `umount(u)`, then `reboot(b)` when my system locks. It's saved me many a fsck. Again, the unmount (remount read-only) hasn't taken place until you see the "OK" and "Done" message appear on the screen.

The loglevels `0`-`9` are useful when your console is being flooded with kernel messages you do not want to see. Selecting `0` will prevent all but the most urgent kernel messages from reaching your console. (They will still be logged if syslogd/klogd are alive, though.)

`term(e)` and `kill(i)` are useful if you have some sort of runaway process you are unable to kill any other way, especially if it's spawning other processes.

"just thaw `it(j)`" is useful if your system becomes unresponsive due to a frozen (probably root) filesystem via the FIFREEZE ioctl.

Sometimes SysRq seems to get 'stuck' after using it, what can I do?

That happens to me, also. I've found that tapping shift, alt, and control on both sides of the keyboard, and hitting an invalid sysrq sequence again will fix the problem. (i.e., something like `alt-sysrq-z`). Switching to another virtual console (ALT+Fn) and then back again should also help.

I hit SysRq, but nothing seems to happen, what's wrong?

There are some keyboards that produce a different keycode for SysRq than the pre-defined value of 99 (see KEY_SYSRQ in `include/uapi/linux/input-event-codes.h`), or which don't have a SysRq key at all. In these cases, run `showkey -s` to find an appropriate scancode sequence, and use `setkeycodes <sequence> 99` to map this sequence to the usual SysRq code (e.g., `setkeycodes e05b 99`). It's probably best to put this command in a boot script. Oh, and by the way, you exit `showkey` by not typing anything for ten seconds.

I want to add SysRQ key events to a module, how does it work?

In order to register a basic function with the table, you must first include the header `include/linux/sysrq.h`, this will define everything else you need. Next, you must create a `sysrq_key_op` struct, and populate it with A) the key handler function you will use, B) a help_msg string, that will print when SysRQ prints help, and C) an action_msg string, that will print right before your handler is called. Your handler must conform to the prototype in 'sysrq.h'.

After the `sysrq_key_op` is created, you can call the kernel function `register_sysrq_key(int key,struct sysrq_key_op *op_p);` this will register the operation pointed to by `op_p` at table key 'key', if that slot in the table is blank. At module unload time, you must call the function `unregister_sysrq_key(int key,struct sysrq_key_op *op_p)`, which will remove the key op pointed to by 'op_p' from the key 'key', if and only if it is currently registered in that slot. This is in case the slot has been overwritten since you registered it.

The Magic SysRQ system works by registering key operations against a key op lookup table, which is defined in 'drivers/tty/sysrq.c'. This key table has a number of operations registered into it at compile time, but is mutable, and 2 functions are exported for interface to it:

```
register_sysrq_key and unregister_sysrq_key.
```

Of course, never ever leave an invalid pointer in the table. I.e., when your module that called register_sysrq_key() exits, it must call unregister_sysrq_key() to clean up the sysrq key table entry that it used. Null pointers in the table are always safe. :)

If for some reason you feel the need to call the handle_sysrq function from within a function called by handle_sysrq, you must be aware that you are in a lock (you are also in an interrupt handler, which means don't sleep!), so you must call `__handle_sysrq_nolock` instead.

When I hit a SysRq key combination only the header appears on the console?

Sysrq output is subject to the same console loglevel control as all other console output. This means that if the kernel was booted 'quiet' as is common on distro kernels the output may not appear on the actual console, even though it will appear in the dmesg buffer, and be accessible via the dmesg command and to the consumers of `/proc/kmsg`. As a specific exception the header line from the sysrq command is passed to all console consumers as if the current loglevel was maximum. If only the header is emitted it is almost certain that the kernel loglevel is too low. Should you require the output on the console channel then you will need to temporarily up the console loglevel using `alt-sysrq-8` or:

```
echo 8 > /proc/sysrq-trigger
```

Remember to return the loglevel to normal after triggering the sysrq command you are interested in.

I have more questions, who can I ask?

Just ask them on the linux-kernel mailing list: linux-kernel@vger.kernel.org

Credits

Written by Mydraal <vulpyne@vulpyne.net> Updated by Adam Sulmicki <adam@cfar.umd.edu> Updated by Jeremy M. Dolan <jmd@turbogeek.org> 2001/01/28 10:15:59 Added to by Crutcher Dunnavant <crutcher+kernel@datastacks.com>

CHAPTER

TWENTY

UNICODE SUPPORT

Last update: 2005-01-17, version 1.4

This file is maintained by H. Peter Anvin <unicode@lanana.org> as part of the Linux Assigned Names And Numbers Authority (LANANA) project. The current version can be found at:

http://www.lanana.org/docs/unicode/admin-guide/unicode.rst

Introduction

The Linux kernel code has been rewritten to use Unicode to map characters to fonts. By downloading a single Unicode-to-font table, both the eight-bit character sets and UTF-8 mode are changed to use the font as indicated.

This changes the semantics of the eight-bit character tables subtly. The four character tables are now:

Map symbol	Map name	Escape code (G0)
LAT1_MAP	Latin-1 (ISO 8859-1)	ESC (B
GRAF_MAP	DEC VT100 pseudographics	ESC (0
IBMPC_MAP	IBM code page 437	ESC (U
USER_MAP	User defined	ESC (K

In particular, ESC (U is no longer “straight to font”, since the font might be completely different than the IBM character set. This permits for example the use of block graphics even with a Latin-1 font loaded.

Note that although these codes are similar to ISO 2022, neither the codes nor their uses match ISO 2022; Linux has two 8-bit codes (G0 and G1), whereas ISO 2022 has four 7-bit codes (G0-G3).

In accordance with the Unicode standard/ISO 10646 the range U+F000 to U+F8FF has been reserved for OS-wide allocation (the Unicode Standard refers to this as a “Corporate Zone”, since this is inaccurate for Linux we call it the “Linux Zone”). U+F000 was picked as the starting point since it lets the direct-mapping area start on a large power of two (in case 1024- or 2048-character fonts ever become necessary). This leaves U+E000 to U+EFFF as End User Zone.

[v1.2]: The Unicodes range from U+F000 and up to U+F7FF have been hard-coded to map directly to the loaded font, bypassing the translation table. The user-defined map now defaults to U+F000 to U+F0FF, emulating the previous behaviour. In practice, this range might be shorter; for example, vgacon can only handle 256-character (U+F000..U+F0FF) or 512-character (U+F000..U+F1FF) fonts.

Actual characters assigned in the Linux Zone

In addition, the following characters not present in Unicode 1.1.4 have been defined; these are used by the DEC VT graphics map. [v1.2] THIS USE IS OBSOLETE AND SHOULD NO LONGER BE USED; PLEASE SEE BELOW.

U+F800	DEC VT GRAPHICS HORIZONTAL LINE SCAN 1
U+F801	DEC VT GRAPHICS HORIZONTAL LINE SCAN 3
U+F803	DEC VT GRAPHICS HORIZONTAL LINE SCAN 7
U+F804	DEC VT GRAPHICS HORIZONTAL LINE SCAN 9

The DEC VT220 uses a 6x10 character matrix, and these characters form a smooth progression in the DEC VT graphics character set. I have omitted the scan 5 line, since it is also used as a block-graphics character, and hence has been coded as U+2500 FORMS LIGHT HORIZONTAL.

[v1.3]: These characters have been officially added to Unicode 3.2.0; they are added at U+23BA, U+23BB, U+23BC, U+23BD. Linux now uses the new values.

[v1.2]: The following characters have been added to represent common keyboard symbols that are unlikely to ever be added to Unicode proper since they are horribly vendor-specific. This, of course, is an excellent example of horrible design.

U+F810	KEYBOARD SYMBOL FLYING FLAG
U+F811	KEYBOARD SYMBOL PULLDOWN MENU
U+F812	KEYBOARD SYMBOL OPEN APPLE
U+F813	KEYBOARD SYMBOL SOLID APPLE

Klingon language support

In 1996, Linux was the first operating system in the world to add support for the artificial language Klingon, created by Marc Okrand for the “Star Trek” television series. This encoding was later adopted by the ConScript Unicode Registry and proposed (but ultimately rejected) for inclusion in Unicode Plane 1. Thus, it remains as a Linux/CSUR private assignment in the Linux Zone.

This encoding has been endorsed by the Klingon Language Institute. For more information, contact them at:

http://www.kli.org/

Since the characters in the beginning of the Linux CZ have been more of the dingbats/symbols/forms type and this is a language, I have located it at the end, on a 16-cell boundary in keeping with standard Unicode practice.

Note:

This range is now officially managed by the ConScript Unicode Registry. The normative reference is at:

http://www.evertype.com/standards/csur/klingon.html

Klingon has an alphabet of 26 characters, a positional numeric writing system with 10 digits, and is written left-to-right, top-to-bottom.

Several glyph forms for the Klingon alphabet have been proposed. However, since the set of symbols appear to be consistent throughout, with only the actual shapes being different, in keeping with standard Unicode practice these differences are considered font variants.

U+F8D0	KLINGON LETTER A
U+F8D1	KLINGON LETTER B
U+F8D2	KLINGON LETTER CH
U+F8D3	KLINGON LETTER D
U+F8D4	KLINGON LETTER E
U+F8D5	KLINGON LETTER GH
U+F8D6	KLINGON LETTER H
U+F8D7	KLINGON LETTER I

Continued on next page

Table 20.1 – continued from previous page

U+F8D8	KLINGON LETTER J
U+F8D9	KLINGON LETTER L
U+F8DA	KLINGON LETTER M
U+F8DB	KLINGON LETTER N
U+F8DC	KLINGON LETTER NG
U+F8DD	KLINGON LETTER O
U+F8DE	KLINGON LETTER P
U+F8DF	KLINGON LETTER Q - Written <q> in standard Okrand Latin transliteration
U+F8E0	KLINGON LETTER QH - Written <Q> in standard Okrand Latin transliteration
U+F8E1	KLINGON LETTER R
U+F8E2	KLINGON LETTER S
U+F8E3	KLINGON LETTER T
U+F8E4	KLINGON LETTER TLH
U+F8E5	KLINGON LETTER U
U+F8E6	KLINGON LETTER V
U+F8E7	KLINGON LETTER W
U+F8E8	KLINGON LETTER Y
U+F8E9	KLINGON LETTER GLOTTAL STOP
U+F8F0	KLINGON DIGIT ZERO
U+F8F1	KLINGON DIGIT ONE
U+F8F2	KLINGON DIGIT TWO
U+F8F3	KLINGON DIGIT THREE
U+F8F4	KLINGON DIGIT FOUR
U+F8F5	KLINGON DIGIT FIVE
U+F8F6	KLINGON DIGIT SIX
U+F8F7	KLINGON DIGIT SEVEN
U+F8F8	KLINGON DIGIT EIGHT
U+F8F9	KLINGON DIGIT NINE
U+F8FD	KLINGON COMMA
U+F8FE	KLINGON FULL STOP
U+F8FF	KLINGON SYMBOL FOR EMPIRE

Other Fictional and Artificial Scripts

Since the assignment of the Klingon Linux Unicode block, a registry of fictional and artificial scripts has been established by John Cowan <jcowan@reutershealth.com> and Michael Everson <everson@evertype.com>. The ConScript Unicode Registry is accessible at:

http://www.evertype.com/standards/csur/

The ranges used fall at the low end of the End User Zone and can hence not be normatively assigned, but it is recommended that people who wish to encode fictional scripts use these codes, in the interest of interoperability. For Klingon, CSUR has adopted the Linux encoding. The CSUR people are driving adding Tengwar and Cirth into Unicode Plane 1; the addition of Klingon to Unicode Plane 1 has been rejected and so the above encoding remains official.

CHAPTER

TWENTYONE

SOFTWARE CURSOR FOR VGA

by Pavel Machek <pavel@atrey.karlin.mff.cuni.cz> and Martin Mares <mj@atrey.karlin.mff.cuni.cz>

Linux now has some ability to manipulate cursor appearance. Normally, you can set the size of hardware cursor. You can now play a few new tricks: you can make your cursor look like a non-blinking red block, make it inverse background of the character it's over or to highlight that character and still choose whether the original hardware cursor should remain visible or not. There may be other things I have never thought of.

The cursor appearance is controlled by a <ESC>[?1;2;3c escape sequence where 1, 2 and 3 are parameters described below. If you omit any of them, they will default to zeroes.

first Parameter specifies cursor size:

```
0=default
1=invisible
2=underline,
...
8=full block
+ 16 if you want the software cursor to be applied
+ 32 if you want to always change the background color
+ 64 if you dislike having the background the same as the
     foreground.
```

Highlights are ignored for the last two flags.

second parameter selects character attribute bits you want to change (by simply XORing them with the value of this parameter). On standard VGA, the high four bits specify background and the low four the foreground. In both groups, low three bits set color (as in normal color codes used by the console) and the most significant one turns on highlight (or sometimes blinking – it depends on the configuration of your VGA).

third parameter consists of character attribute bits you want to set.

Bit setting takes place before bit toggling, so you can simply clear a bit by including it in both the set mask and the toggle mask.

Examples

To get normal blinking underline, use:

```
echo -e '\033[?2c'
```

To get blinking block, use:

```
echo -e '\033[?6c'
```

To get red non-blinking block, use:

```
echo -e '\033[?17;0;64c'
```

CHAPTER

TWENTYTWO

KERNEL SUPPORT FOR MISCELLANEOUS (YOUR FAVOURITE) BINARY FORMATS V1.1

This Kernel feature allows you to invoke almost (for restrictions see below) every program by simply typing its name in the shell. This includes for example compiled Java(TM), Python or Emacs programs.

To achieve this you must tell binfmt_misc which interpreter has to be invoked with which binary. Binfmt_misc recognises the binary-type by matching some bytes at the beginning of the file with a magic byte sequence (masking out specified bits) you have supplied. Binfmt_misc can also recognise a filename extension aka `.com` or `.exe`.

First you must mount binfmt_misc:

```
mount binfmt_misc -t binfmt_misc /proc/sys/fs/binfmt_misc
```

To actually register a new binary type, you have to set up a string looking like `:name:type:offset:magic:mask:interpreter:flags` (where you can choose the `:` upon your needs) and echo it to `/proc/sys/fs/binfmt_misc/register`.

Here is what the fields mean:

- **`name`** is an identifier string. A new /proc file will be created with this `name` below `/proc/sys/fs/binfmt_misc`; cannot contain slashes `/` for obvious reasons.
- **`type`** is the type of recognition. Give `M` for magic and `E` for extension.
- **`offset`** is the offset of the magic/mask in the file, counted in bytes. This defaults to 0 if you omit it (i.e. you write `:name:type::magic...`). Ignored when using filename extension matching.
- **`magic`** is the byte sequence binfmt_misc is matching for. The magic string may contain hex-encoded characters like `\x0a` or `\xA4`. Note that you must escape any NUL bytes; parsing halts at the first one. In a shell environment you might have to write `\\x0a` to prevent the shell from eating your `\`. If you chose filename extension matching, this is the extension to be recognised (without the `.`, the `\x0a` specials are not allowed). Extension matching is case sensitive, and slashes `/` are not allowed!
- **`mask`** is an (optional, defaults to all 0xff) mask. You can mask out some bits from matching by supplying a string like magic and as long as magic. The mask is anded with the byte sequence of the file. Note that you must escape any NUL bytes; parsing halts at the first one. Ignored when using filename extension matching.
- **`interpreter`** is the program that should be invoked with the binary as first argument (specify the full path)
- **`flags`** is an optional field that controls several aspects of the invocation of the interpreter. It is a string of capital letters, each controls a certain aspect. The following flags are supported:

  P - preserve-argv[0] Legacy behavior of binfmt_misc is to overwrite the original argv[0] with the full path to the binary. When this flag is included, binfmt_misc will add an argument to the argument vector for this purpose, thus preserving the original `argv[0]`. e.g. If your interp is set to `/bin/foo` and you run `blah` (which is in `/usr/local/bin`), then the kernel will execute `/bin/foo` with `argv[]` set to

["/bin/foo","/usr/local/bin/blah","blah"]. The interp has to be aware of this so it can execute /usr/local/bin/blah with argv[] set to ["blah"].

0 - open-binary Legacy behavior of binfmt_misc is to pass the full path of the binary to the interpreter as an argument. When this flag is included, binfmt_misc will open the file for reading and pass its descriptor as an argument, instead of the full path, thus allowing the interpreter to execute non-readable binaries. This feature should be used with care - the interpreter has to be trusted not to emit the contents of the non-readable binary.

C - credentials Currently, the behavior of binfmt_misc is to calculate the credentials and security token of the new process according to the interpreter. When this flag is included, these attributes are calculated according to the binary. It also implies the 0 flag. This feature should be used with care as the interpreter will run with root permissions when a setuid binary owned by root is run with binfmt_misc.

F - fix binary The usual behaviour of binfmt_misc is to spawn the binary lazily when the misc format file is invoked. However, this doesn''t work very well in the face of mount namespaces and changeroots, so the F mode opens the binary as soon as the emulation is installed and uses the opened image to spawn the emulator, meaning it is always available once installed, regardless of how the environment changes.

There are some restrictions:

- the whole register string may not exceed 1920 characters
- the magic must reside in the first 128 bytes of the file, i.e. offset+size(magic) has to be less than 128
- the interpreter string may not exceed 127 characters

To use binfmt_misc you have to mount it first. You can mount it with mount -t binfmt_misc none /proc/sys/fs/binfmt_misc command, or you can add a line none /proc/sys/fs/binfmt_misc binfmt_misc defaults 0 0 to your /etc/fstab so it auto mounts on boot.

You may want to add the binary formats in one of your /etc/rc scripts during boot-up. Read the manual of your init program to figure out how to do this right.

Think about the order of adding entries! Later added entries are matched first!

A few examples (assumed you are in /proc/sys/fs/binfmt_misc):

- enable support for em86 (like binfmt_em86, for Alpha AXP only):

```
echo ':i386:M::\x7fELF\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x03:
 ↪\xff\xff\xff\xff\xff\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfb\xff\xff:/bin/em86:'␣
 ↪> register
echo ':i486:M::\x7fELF\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x06:
 ↪\xff\xff\xff\xff\xff\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfb\xff\xff:/bin/em86:'␣
 ↪> register
```

- enable support for packed DOS applications (pre-configured dosemu hdimages):

```
echo ':DEXE:M::\x0eDEX::/usr/bin/dosexec:' > register
```

- enable support for Windows executables using wine:

```
echo ':DOSWin:M::MZ::/usr/local/bin/wine:' > register
```

For java support see Documentation/admin-guide/java.rst

You can enable/disable binfmt_misc or one binary type by echoing 0 (to disable) or 1 (to enable) to /proc/sys/fs/binfmt_misc/status or /proc/.../the_name. Catting the file tells you the current status of binfmt_misc/the_entry.

You can remove one entry or all entries by echoing -1 to /proc/.../the_name or /proc/sys/fs/binfmt_misc/status.

Hints

If you want to pass special arguments to your interpreter, you can write a wrapper script for it. See Documentation/admin-guide/java.rst for an example.

Your interpreter should NOT look in the PATH for the filename; the kernel passes it the full filename (or the file descriptor) to use. Using $PATH can cause unexpected behaviour and can be a security hazard.

Richard Günther <rguenth@tat.physik.uni-tuebingen.de>

CHAPTER

TWENTYTHREE

MONO(TM) BINARY KERNEL SUPPORT FOR LINUX

To configure Linux to automatically execute Mono-based .NET binaries (in the form of .exe files) without the need to use the mono CLR wrapper, you can use the BINFMT_MISC kernel support.

This will allow you to execute Mono-based .NET binaries just like any other program after you have done the following:

1. You MUST FIRST install the Mono CLR support, either by downloading a binary package, a source tarball or by installing from CVS. Binary packages for several distributions can be found at:

   http://go-mono.com/download.html

   Instructions for compiling Mono can be found at:

   http://www.go-mono.com/compiling.html

   Once the Mono CLR support has been installed, just check that `/usr/bin/mono` (which could be located elsewhere, for example `/usr/local/bin/mono`) is working.

2. You have to compile BINFMT_MISC either as a module or into the kernel (`CONFIG_BINFMT_MISC`) and set it up properly. If you choose to compile it as a module, you will have to insert it manually with modprobe/insmod, as kmod cannot be easily supported with binfmt_misc. Read the file `binfmt_misc.txt` in this directory to know more about the configuration process.

3. Add the following entries to `/etc/rc.local` or similar script to be run at system startup:

```
# Insert BINFMT_MISC module into the kernel
if [ ! -e /proc/sys/fs/binfmt_misc/register ]; then
    /sbin/modprobe binfmt_misc
    # Some distributions, like Fedora Core, perform
    # the following command automatically when the
    # binfmt_misc module is loaded into the kernel
    # or during normal boot up (systemd-based systems).
    # Thus, it is possible that the following line
    # is not needed at all.
    mount -t binfmt_misc none /proc/sys/fs/binfmt_misc
fi

# Register support for .NET CLR binaries
if [ -e /proc/sys/fs/binfmt_misc/register ]; then
    # Replace /usr/bin/mono with the correct pathname to
    # the Mono CLR runtime (usually /usr/local/bin/mono
    # when compiling from sources or CVS).
    echo ':CLR:M::MZ::/usr/bin/mono:' > /proc/sys/fs/binfmt_misc/register
else
    echo "No binfmt_misc support"
    exit 1
fi
```

4. Check that `.exe` binaries can be ran without the need of a wrapper script, simply by launching the `.exe` file directly from a command prompt, for example:

```
/usr/bin/xsd.exe
```

Note:

If this fails with a permission denied error, check that the .exe file has execute permissions.

CHAPTER

TWENTYFOUR

JAVA(TM) BINARY KERNEL SUPPORT FOR LINUX V1.03

Linux beats them ALL! While all other OS's are TALKING about direct support of Java Binaries in the OS, Linux is doing it!

You can execute Java applications and Java Applets just like any other program after you have done the following:

1. You MUST FIRST install the Java Developers Kit for Linux. The Java on Linux HOWTO gives the details on getting and installing this. This HOWTO can be found at:

   ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/Java-HOWTO

   You should also set up a reasonable CLASSPATH environment variable to use Java applications that make use of any nonstandard classes (not included in the same directory as the application itself).

2. You have to compile BINFMT_MISC either as a module or into the kernel (`CONFIG_BINFMT_MISC`) and set it up properly. If you choose to compile it as a module, you will have to insert it manually with modprobe/insmod, as kmod cannot easily be supported with binfmt_misc. Read the file 'binfmt_misc.txt' in this directory to know more about the configuration process.

3. Add the following configuration items to binfmt_misc (you should really have read `binfmt_misc.txt` now): support for Java applications:

   ```
   ':Java:M::\xca\xfe\xba\xbe::/usr/local/bin/javawrapper:'
   ```

   support for executable Jar files:

   ```
   ':ExecutableJAR:E::jar::/usr/local/bin/jarwrapper:'
   ```

   support for Java Applets:

   ```
   ':Applet:E::html::/usr/bin/appletviewer:'
   ```

   or the following, if you want to be more selective:

   ```
   ':Applet:M::<!--applet::/usr/bin/appletviewer:'
   ```

   Of course you have to fix the path names. The path/file names given in this document match the Debian 2.1 system. (i.e. jdk installed in `/usr`, custom wrappers from this document in `/usr/local`)

   Note, that for the more selective applet support you have to modify existing html-files to contain `<!--applet-->` in the first line (< has to be the first character!) to let this work!

   For the compiled Java programs you need a wrapper script like the following (this is because Java is broken in case of the filename handling), again fix the path names, both in the script and in the above given configuration string.

   You, too, need the little program after the script. Compile like:

   ```
   gcc -O2 -o javaclassname javaclassname.c
   ```

and stick it to /usr/local/bin.

Both the javawrapper shellscript and the javaclassname program were supplied by Colin J. Watson <cjw44@cam.ac.uk>.

Javawrapper shell script:

```
#!/bin/bash
# /usr/local/bin/javawrapper - the wrapper for binfmt_misc/java

if [ -z "$1" ]; then
      exec 1>&2
      echo Usage: $0 class-file
      exit 1
fi

CLASS=$1
FQCLASS=`/usr/local/bin/javaclassname $1`
FQCLASSN=`echo $FQCLASS | sed -e 's/^.*\.\([^.]*\)$/\1/'`
FQCLASSP=`echo $FQCLASS | sed -e 's-\.-/-g' -e 's-^[^/]*$--' -e 's-/[^/]*$--'`

# for example:
# CLASS=Test.class
# FQCLASS=foo.bar.Test
# FQCLASSN=Test
# FQCLASSP=foo/bar

unset CLASSBASE

declare -i LINKLEVEL=0

while :; do
      if [ "`basename $CLASS .class`" == "$FQCLASSN" ]; then
              # See if this directory works straight off
              cd -L `dirname $CLASS`
              CLASSDIR=$PWD
              cd $OLDPWD
              if echo $CLASSDIR | grep -q "$FQCLASSP$"; then
                      CLASSBASE=`echo $CLASSDIR | sed -e "s.$FQCLASSP$.."`
                      break;
              fi
              # Try dereferencing the directory name
              cd -P `dirname $CLASS`
              CLASSDIR=$PWD
              cd $OLDPWD
              if echo $CLASSDIR | grep -q "$FQCLASSP$"; then
                      CLASSBASE=`echo $CLASSDIR | sed -e "s.$FQCLASSP$.."`
                      break;
              fi
              # If no other possible filename exists
              if [ ! -L $CLASS ]; then
                      exec 1>&2
                      echo $0:
                      echo "  $CLASS should be in a" \
                           "directory tree called $FQCLASSP"
                      exit 1
              fi
      fi
      if [ ! -L $CLASS ]; then break; fi
      # Go down one more level of symbolic links
      let LINKLEVEL+=1
      if [ $LINKLEVEL -gt 5 ]; then
              exec 1>&2
              echo $0:
```

```
                echo "  Too many symbolic links encountered"
                exit 1
        fi
        CLASS=`ls --color=no -l $CLASS | sed -e 's/^.* \([^ ]*\)$/\1/'`
done

if [ -z "$CLASSBASE" ]; then
        if [ -z "$FQCLASSP" ]; then
                GOODNAME=$FQCLASSN.class
        else
                GOODNAME=$FQCLASSP/$FQCLASSN.class
        fi
        exec 1>&2
        echo $0:
        echo "  $FQCLASS should be in a file called $GOODNAME"
        exit 1
fi

if ! echo $CLASSPATH | grep -q "^\(.*:\)*$CLASSBASE\(:.*\)*"; then
        # class is not in CLASSPATH, so prepend dir of class to CLASSPATH
        if [ -z "${CLASSPATH}" ] ; then
                export CLASSPATH=$CLASSBASE
        else
                export CLASSPATH=$CLASSBASE:$CLASSPATH
        fi
fi

shift
/usr/bin/java $FQCLASS "$@"
```

javaclassname.c:

```
/* javaclassname.c
 *
 * Extracts the class name from a Java class file; intended for use in a Java
 * wrapper of the type supported by the binfmt_misc option in the Linux kernel.
 *
 * Copyright (C) 1999 Colin J. Watson <cjw44@cam.ac.uk>.
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <sys/types.h>

/* From Sun's Java VM Specification, as tag entries in the constant pool. */

#define CP_UTF8 1
#define CP_INTEGER 3
```

```
#define CP_FLOAT 4
#define CP_LONG 5
#define CP_DOUBLE 6
#define CP_CLASS 7
#define CP_STRING 8
#define CP_FIELDREF 9
#define CP_METHODREF 10
#define CP_INTERFACEMETHODREF 11
#define CP_NAMEANDTYPE 12
#define CP_METHODHANDLE 15
#define CP_METHODTYPE 16
#define CP_INVOKEDYNAMIC 18

/* Define some commonly used error messages */

#define seek_error() error("%s: Cannot seek\n", program)
#define corrupt_error() error("%s: Class file corrupt\n", program)
#define eof_error() error("%s: Unexpected end of file\n", program)
#define utf8_error() error("%s: Only ASCII 1-255 supported\n", program);

char *program;

long *pool;

u_int8_t read_8(FILE *classfile);
u_int16_t read_16(FILE *classfile);
void skip_constant(FILE *classfile, u_int16_t *cur);
void error(const char *format, ...);
int main(int argc, char **argv);

/* Reads in an unsigned 8-bit integer. */
u_int8_t read_8(FILE *classfile)
{
      int b = fgetc(classfile);
      if(b == EOF)
              eof_error();
      return (u_int8_t)b;
}

/* Reads in an unsigned 16-bit integer. */
u_int16_t read_16(FILE *classfile)
{
      int b1, b2;
      b1 = fgetc(classfile);
      if(b1 == EOF)
              eof_error();
      b2 = fgetc(classfile);
      if(b2 == EOF)
              eof_error();
      return (u_int16_t)((b1 << 8) | b2);
}

/* Reads in a value from the constant pool. */
void skip_constant(FILE *classfile, u_int16_t *cur)
{
      u_int16_t len;
      int seekerr = 1;
      pool[*cur] = ftell(classfile);
      switch(read_8(classfile))
      {
      case CP_UTF8:
              len = read_16(classfile);
              seekerr = fseek(classfile, len, SEEK_CUR);
```

```
                break;
        case CP_CLASS:
        case CP_STRING:
        case CP_METHODTYPE:
                seekerr = fseek(classfile, 2, SEEK_CUR);
                break;
        case CP_METHODHANDLE:
                seekerr = fseek(classfile, 3, SEEK_CUR);
                break;
        case CP_INTEGER:
        case CP_FLOAT:
        case CP_FIELDREF:
        case CP_METHODREF:
        case CP_INTERFACEMETHODREF:
        case CP_NAMEANDTYPE:
        case CP_INVOKEDYNAMIC:
                seekerr = fseek(classfile, 4, SEEK_CUR);
                break;
        case CP_LONG:
        case CP_DOUBLE:
                seekerr = fseek(classfile, 8, SEEK_CUR);
                ++(*cur);
                break;
        default:
                corrupt_error();
        }
        if(seekerr)
                seek_error();
}

void error(const char *format, ...)
{
        va_list ap;
        va_start(ap, format);
        vfprintf(stderr, format, ap);
        va_end(ap);
        exit(1);
}

int main(int argc, char **argv)
{
        FILE *classfile;
        u_int16_t cp_count, i, this_class, classinfo_ptr;
        u_int8_t length;

        program = argv[0];

        if(!argv[1])
                error("%s: Missing input file\n", program);
        classfile = fopen(argv[1], "rb");
        if(!classfile)
                error("%s: Error opening %s\n", program, argv[1]);

        if(fseek(classfile, 8, SEEK_SET))  /* skip magic and version numbers */
                seek_error();
        cp_count = read_16(classfile);
        pool = calloc(cp_count, sizeof(long));
        if(!pool)
                error("%s: Out of memory for constant pool\n", program);

        for(i = 1; i < cp_count; ++i)
                skip_constant(classfile, &i);
        if(fseek(classfile, 2, SEEK_CUR))       /* skip access flags */
```

```
                seek_error();

        this_class = read_16(classfile);
        if(this_class < 1 || this_class >= cp_count)
                corrupt_error();
        if(!pool[this_class] || pool[this_class] == -1)
                corrupt_error();
        if(fseek(classfile, pool[this_class] + 1, SEEK_SET))
                seek_error();

        classinfo_ptr = read_16(classfile);
        if(classinfo_ptr < 1 || classinfo_ptr >= cp_count)
                corrupt_error();
        if(!pool[classinfo_ptr] || pool[classinfo_ptr] == -1)
                corrupt_error();
        if(fseek(classfile, pool[classinfo_ptr] + 1, SEEK_SET))
                seek_error();

        length = read_16(classfile);
        for(i = 0; i < length; ++i)
        {
                u_int8_t x = read_8(classfile);
                if((x & 0x80) || !x)
                {
                        if((x & 0xE0) == 0xC0)
                        {
                                u_int8_t y = read_8(classfile);
                                if((y & 0xC0) == 0x80)
                                {
                                        int c = ((x & 0x1f) << 6) + (y & 0x3f);
                                        if(c) putchar(c);
                                        else utf8_error();
                                }
                                else utf8_error();
                        }
                        else utf8_error();
                }
                else if(x == '/') putchar('.');
                else putchar(x);
        }
        putchar('\n');
        free(pool);
        fclose(classfile);
        return 0;
}
```

jarwrapper:

```
#!/bin/bash
# /usr/local/java/bin/jarwrapper - the wrapper for binfmt_misc/jar

java -jar $1
```

Now simply `chmod +x` the `.class`, `.jar` and/or `.html` files you want to execute.

To add a Java program to your path best put a symbolic link to the main .class file into /usr/bin (or another place you like) omitting the .class extension. The directory containing the original .class file will be added to your CLASSPATH during execution.

To test your new setup, enter in the following simple Java app, and name it “HelloWorld.java”:

```
class HelloWorld {
        public static void main(String args[]) {
```

```
                System.out.println("Hello World!");
        }
}
```

Now compile the application with:

```
javac HelloWorld.java
```

Set the executable permissions of the binary file, with:

```
chmod 755 HelloWorld.class
```

And then execute it:

```
./HelloWorld.class
```

To execute Java Jar files, simple chmod the `*.jar` files to include the execution bit, then just do:

```
./Application.jar
```

To execute Java Applets, simple chmod the `*.html` files to include the execution bit, then just do:

```
./Applet.html
```

originally by Brian A. Lantz, brian@lantz.com heavily edited for binfmt_misc by Richard Günther new scripts by Colin J. Watson <cjw44@cam.ac.uk> added executable Jar file support by Kurt Huwig <kurt@iku-netz.de>

CHAPTER

TWENTYFIVE

RELIABILITY, AVAILABILITY AND SERVICEABILITY

RAS concepts

Reliability, Availability and Serviceability (RAS) is a concept used on servers meant to measure their robustness.

Reliability is the probability that a system will produce correct outputs.

- Generally measured as Mean Time Between Failures (MTBF)
- Enhanced by features that help to avoid, detect and repair hardware faults

Availability is the probability that a system is operational at a given time

- Generally measured as a percentage of downtime per a period of time
- Often uses mechanisms to detect and correct hardware faults in runtime;

Serviceability (or maintainability) is the simplicity and speed with which a system can be repaired or maintained

- Generally measured on Mean Time Between Repair (MTBR)

Improving RAS

In order to reduce systems downtime, a system should be capable of detecting hardware errors, and, when possible correcting them in runtime. It should also provide mechanisms to detect hardware degradation, in order to warn the system administrator to take the action of replacing a component before it causes data loss or system downtime.

Among the monitoring measures, the most usual ones include:

- CPU – detect errors at instruction execution and at L1/L2/L3 caches;
- Memory – add error correction logic (ECC) to detect and correct errors;
- I/O – add CRC checksums for transferred data;
- Storage – RAID, journal file systems, checksums, Self-Monitoring, Analysis and Reporting Technology (SMART).

By monitoring the number of occurrences of error detections, it is possible to identify if the probability of hardware errors is increasing, and, on such case, do a preventive maintenance to replace a degraded component while those errors are correctable.

Types of errors

Most mechanisms used on modern systems use use technologies like Hamming Codes that allow error correction when the number of errors on a bit packet is below a threshold. If the number of errors is above,

those mechanisms can indicate with a high degree of confidence that an error happened, but they can't correct.

Also, sometimes an error occur on a component that it is not used. For example, a part of the memory that it is not currently allocated.

That defines some categories of errors:

- **Correctable Error (CE)** - the error detection mechanism detected and corrected the error. Such errors are usually not fatal, although some Kernel mechanisms allow the system administrator to consider them as fatal.
- **Uncorrected Error (UE)** - the amount of errors happened above the error correction threshold, and the system was unable to auto-correct.
- **Fatal Error** - when an UE error happens on a critical component of the system (for example, a piece of the Kernel got corrupted by an UE), the only reliable way to avoid data corruption is to hang or reboot the machine.
- **Non-fatal Error** - when an UE error happens on an unused component, like a CPU in power down state or an unused memory bank, the system may still run, eventually replacing the affected hardware by a hot spare, if available.

  Also, when an error happens on a userspace process, it is also possible to kill such process and let userspace restart it.

The mechanism for handling non-fatal errors is usually complex and may require the help of some userspace application, in order to apply the policy desired by the system administrator.

Identifying a bad hardware component

Just detecting a hardware flaw is usually not enough, as the system needs to pinpoint to the minimal replaceable unit (MRU) that should be exchanged to make the hardware reliable again.

So, it requires not only error logging facilities, but also mechanisms that will translate the error message to the silkscreen or component label for the MRU.

Typically, it is very complex for memory, as modern CPUs interlace memory from different memory modules, in order to provide a better performance. The DMI BIOS usually have a list of memory module labels, with can be obtained using the `dmidecode` tool. For example, on a desktop machine, it shows:

```
Memory Device
        Total Width: 64 bits
        Data Width: 64 bits
        Size: 16384 MB
        Form Factor: SODIMM
        Set: None
        Locator: ChannelA-DIMM0
        Bank Locator: BANK 0
        Type: DDR4
        Type Detail: Synchronous
        Speed: 2133 MHz
        Rank: 2
        Configured Clock Speed: 2133 MHz
```

On the above example, a DDR4 SO-DIMM memory module is located at the system's memory labeled as "BANK 0", as given by the *bank locator* field. Please notice that, on such system, the *total width* is equal to the *data width*. It means that such memory module doesn't have error detection/correction mechanisms.

Unfortunately, not all systems use the same field to specify the memory bank. On this example, from an older server, `dmidecode` shows:

```
Memory Device
        Array Handle: 0x1000
```

```
        Error Information Handle: Not Provided
        Total Width: 72 bits
        Data Width: 64 bits
        Size: 8192 MB
        Form Factor: DIMM
        Set: 1
        Locator: DIMM_A1
        Bank Locator: Not Specified
        Type: DDR3
        Type Detail: Synchronous Registered (Buffered)
        Speed: 1600 MHz
        Rank: 2
        Configured Clock Speed: 1600 MHz
```

There, the DDR3 RDIMM memory module is located at the system's memory labeled as "DIMM_A1", as given by the *locator* field. Please notice that this memory module has 64 bits of *data width* and 72 bits of *total width*. So, it has 8 extra bits to be used by error detection and correction mechanisms. Such kind of memory is called Error-correcting code memory (ECC memory).

To make things even worse, it is not uncommon that systems with different labels on their system's board to use exactly the same BIOS, meaning that the labels provided by the BIOS won't match the real ones.

ECC memory

As mentioned on the previous section, ECC memory has extra bits to be used for error correction. So, on 64 bit systems, a memory module has 64 bits of *data width*, and 74 bits of *total width*. So, there are 8 bits extra bits to be used for the error detection and correction mechanisms. Those extra bits are called *syndrome*[1][2].

So, when the cpu requests the memory controller to write a word with *data width*, the memory controller calculates the *syndrome* in real time, using Hamming code, or some other error correction code, like SECDED+, producing a code with *total width* size. Such code is then written on the memory modules.

At read, the *total width* bits code is converted back, using the same ECC code used on write, producing a word with *data width* and a *syndrome*. The word with *data width* is sent to the CPU, even when errors happen.

The memory controller also looks at the *syndrome* in order to check if there was an error, and if the ECC code was able to fix such error. If the error was corrected, a Corrected Error (CE) happened. If not, an Uncorrected Error (UE) happened.

The information about the CE/UE errors is stored on some special registers at the memory controller and can be accessed by reading such registers, either by BIOS, by some special CPUs or by Linux EDAC driver. On x86 64 bit CPUs, such errors can also be retrieved via the Machine Check Architecture (MCA)[3].

[1] Please notice that several memory controllers allow operation on a mode called "Lock-Step", where it groups two memory modules together, doing 128-bit reads/writes. That gives 16 bits for error correction, with significantly improves the error correction mechanism, at the expense that, when an error happens, there's no way to know what memory module is to blame. So, it has to blame both memory modules.

[2] Some memory controllers also allow using memory in mirror mode. On such mode, the same data is written to two memory modules. At read, the system checks both memory modules, in order to check if both provide identical data. On such configuration, when an error happens, there's no way to know what memory module is to blame. So, it has to blame both memory modules (or 4 memory modules, if the system is also on Lock-step mode).

[3] For more details about the Machine Check Architecture (MCA), please read Documentation/x86/x86_64/machinecheck at the Kernel tree.

EDAC - Error Detection And Correction

Note:

"bluesmoke" was the name for this device driver subsystem when it was "out-of-tree" and maintained at http://bluesmoke.sourceforge.net. That site is mostly archaic now and can be used only for historical purposes.
When the subsystem was pushed upstream for the first time, on Kernel 2.6.16, for the first time, it was renamed to EDAC.

Purpose

The edac kernel module's goal is to detect and report hardware errors that occur within the computer system running under linux.

Memory

Memory Correctable Errors (CE) and Uncorrectable Errors (UE) are the primary errors being harvested. These types of errors are harvested by the edac_mc device.

Detecting CE events, then harvesting those events and reporting them, **can** but must not necessarily be a predictor of future UE events. With CE events only, the system can and will continue to operate as no data has been damaged yet.

However, preventive maintenance and proactive part replacement of memory modules exhibiting CEs can reduce the likelihood of the dreaded UE events and system panics.

Other hardware elements

A new feature for EDAC, the edac_device class of device, was added in the 2.6.23 version of the kernel.

This new device type allows for non-memory type of ECC hardware detectors to have their states harvested and presented to userspace via the sysfs interface.

Some architectures have ECC detectors for L1, L2 and L3 caches, along with DMA engines, fabric switches, main data path switches, interconnections, and various other hardware data paths. If the hardware reports it, then a edac_device device probably can be constructed to harvest and present that to userspace.

PCI bus scanning

In addition, PCI devices are scanned for PCI Bus Parity and SERR Errors in order to determine if errors are occurring during data transfers.

The presence of PCI Parity errors must be examined with a grain of salt. There are several add-in adapters that do **not** follow the PCI specification with regards to Parity generation and reporting. The specification says the vendor should tie the parity status bits to 0 if they do not intend to generate parity. Some vendors do not do this, and thus the parity bit can "float" giving false positives.

There is a PCI device attribute located in sysfs that is checked by the EDAC PCI scanning code. If that attribute is set, PCI parity/error scanning is skipped for that device. The attribute is:

```
broken_parity_status
```

and is located in /sys/devices/pci<XXX>/0000:XX:YY.Z directories for PCI devices.

Versioning

EDAC is composed of a "core" module (edac_core.ko) and several Memory Controller (MC) driver modules. On a given system, the CORE is loaded and one MC driver will be loaded. Both the CORE and the MC driver (or edac_device driver) have individual versions that reflect current release level of their respective modules.

Thus, to "report" on what version a system is running, one must report both the CORE's and the MC driver's versions.

Loading

If edac was statically linked with the kernel then no loading is necessary. If edac was built as modules then simply modprobe the edac pieces that you need. You should be able to modprobe hardware-specific modules and have the dependencies load the necessary core modules.

Example:

```
$ modprobe amd76x_edac
```

loads both the amd76x_edac.ko memory controller module and the edac_mc.ko core module.

Sysfs interface

EDAC presents a sysfs interface for control and reporting purposes. It lives in the /sys/devices/system/edac directory.

Within this directory there currently reside 2 components:

mc	memory controller(s) system
pci	PCI control and status system

Memory Controller (mc) Model

Each mc device controls a set of memory modules [4]. These modules are laid out in a Chip-Select Row (csrowX) and Channel table (chX). There can be multiple csrows and multiple channels.

Memory controllers allow for several csrows, with 8 csrows being a typical value. Yet, the actual number of csrows depends on the layout of a given motherboard, memory controller and memory module characteristics.

Dual channels allow for dual data length (e. g. 128 bits, on 64 bit systems) data transfers to/from the CPU from/to memory. Some newer chipsets allow for more than 2 channels, like Fully Buffered DIMMs (FB-DIMMs) memory controllers. The following example will assume 2 channels:

CS Rows	Channels	
	ch0	ch1
csrow0	DIMM_A0	DIMM_B0
csrow1		
csrow2	DIMM_A1	DIMM_B1
csrow3		

In the above example, there are 4 physical slots on the motherboard for memory DIMMs:

DIMM_A0	DIMM_B0
DIMM_A1	DIMM_B1

[4] Nowadays, the term DIMM (Dual In-line Memory Module) is widely used to refer to a memory module, although there are other memory packaging alternatives, like SO-DIMM, SIMM, etc. Along this document, and inside the EDAC system, the term "dimm" is used for all memory modules, even when they use a different kind of packaging.

Labels for these slots are usually silk-screened on the motherboard. Slots labeled A are channel 0 in this example. Slots labeled B are channel 1. Notice that there are two csrows possible on a physical DIMM. These csrows are allocated their csrow assignment based on the slot into which the memory DIMM is placed. Thus, when 1 DIMM is placed in each Channel, the csrows cross both DIMMs.

Memory DIMMs come single or dual "ranked". A rank is a populated csrow. Thus, 2 single ranked DIMMs, placed in slots DIMM_A0 and DIMM_B0 above will have just one csrow (csrow0). csrow1 will be empty. On the other hand, when 2 dual ranked DIMMs are similarly placed, then both csrow0 and csrow1 will be populated. The pattern repeats itself for csrow2 and csrow3.

The representation of the above is reflected in the directory tree in EDAC's sysfs interface. Starting in directory `/sys/devices/system/edac/mc`, each memory controller will be represented by its own `mcX` directory, where X is the index of the MC:

```
..../edac/mc/
           |
           |->mc0
           |->mc1
           |->mc2
           ....
```

Under each `mcX` directory each `csrowX` is again represented by a `csrowX`, where X is the csrow index:

```
.../mc/mc0/
        |
        |->csrow0
        |->csrow2
        |->csrow3
        ....
```

Notice that there is no csrow1, which indicates that csrow0 is composed of a single ranked DIMMs. This should also apply in both Channels, in order to have dual-channel mode be operational. Since both csrow2 and csrow3 are populated, this indicates a dual ranked set of DIMMs for channels 0 and 1.

Within each of the `mcX` and `csrowX` directories are several EDAC control and attribute files.

`mcX` directories

In `mcX` directories are EDAC control and attribute files for this X instance of the memory controllers.

For a description of the sysfs API, please see:

Documentation/ABI/testing/sysfs-devices-edac

`dimmX` or `rankX` directories

The recommended way to use the EDAC subsystem is to look at the information provided by the `dimmX` or `rankX` directories [5].

A typical EDAC system has the following structure under `/sys/devices/system/edac/`[6]:

```
/sys/devices/system/edac/
├-- mc
|   ├-- mc0
|   |   ├-- ce_count
|   |   ├-- ce_noinfo_count
```

[5] On some systems, the memory controller doesn't have any logic to identify the memory module. On such systems, the directory is called `rankX` and works on a similar way as the `csrowX` directories. On modern Intel memory controllers, the memory controller identifies the memory modules directly. On such systems, the directory is called `dimmX`.

[6] There are also some `power` directories and `subsystem` symlinks inside the sysfs mapping that are automatically created by the sysfs subsystem. Currently, they serve no purpose.

```
|   |   ├── dimm0
|   |   |   ├── dimm_ce_count
|   |   |   ├── dimm_dev_type
|   |   |   ├── dimm_edac_mode
|   |   |   ├── dimm_label
|   |   |   ├── dimm_location
|   |   |   ├── dimm_mem_type
|   |   |   ├── dimm_ue_count
|   |   |   ├── size
|   |   |   └── uevent
|   |   ├── max_location
|   |   ├── mc_name
|   |   ├── reset_counters
|   |   ├── seconds_since_reset
|   |   ├── size_mb
|   |   ├── ue_count
|   |   ├── ue_noinfo_count
|   |   └── uevent
|   ├── mc1
|   |   ├── ce_count
|   |   ├── ce_noinfo_count
|   |   ├── dimm0
|   |   |   ├── dimm_ce_count
|   |   |   ├── dimm_dev_type
|   |   |   ├── dimm_edac_mode
|   |   |   ├── dimm_label
|   |   |   ├── dimm_location
|   |   |   ├── dimm_mem_type
|   |   |   ├── dimm_ue_count
|   |   |   ├── size
|   |   |   └── uevent
|   |   ├── max_location
|   |   ├── mc_name
|   |   ├── reset_counters
|   |   ├── seconds_since_reset
|   |   ├── size_mb
|   |   ├── ue_count
|   |   ├── ue_noinfo_count
|   |   └── uevent
|   └── uevent
└── uevent
```

In the dimmX directories are EDAC control and attribute files for this X memory module:

- `size` - Total memory managed by this csrow attribute file

  This attribute file displays, in count of megabytes, the memory that this csrow contains.

- `dimm_ue_count` - Uncorrectable Errors count attribute file

  This attribute file displays the total count of uncorrectable errors that have occurred on this DIMM. If panic_on_ue is set this counter will not have a chance to increment, since EDAC will panic the system.

- `dimm_ce_count` - Correctable Errors count attribute file

  This attribute file displays the total count of correctable errors that have occurred on this DIMM. This count is very important to examine. CEs provide early indications that a DIMM is beginning to fail. This count field should be monitored for non-zero values and report such information to the system administrator.

- `dimm_dev_type` - Device type attribute file

This attribute file will display what type of DRAM device is being utilized on this DIMM. Examples:

- x1
- x2
- x4
- x8

- `dimm_edac_mode` - EDAC Mode of operation attribute file

  This attribute file will display what type of Error detection and correction is being utilized.

- `dimm_label` - memory module label control file

  This control file allows this DIMM to have a label assigned to it. With this label in the module, when errors occur the output can provide the DIMM label in the system log. This becomes vital for panic events to isolate the cause of the UE event.

  DIMM Labels must be assigned after booting, with information that correctly identifies the physical slot with its silk screen label. This information is currently very motherboard specific and determination of this information must occur in userland at this time.

- `dimm_location` - location of the memory module

  The location can have up to 3 levels, and describe how the memory controller identifies the location of a memory module. Depending on the type of memory and memory controller, it can be:

  - *csrow* and *channel* - used when the memory controller doesn't identify a single DIMM - e. g. in `rank`X dir;
  - *branch*, *channel*, *slot* - typically used on FB-DIMM memory controllers;
  - *channel*, *slot* - used on Nehalem and newer Intel drivers.

- `dimm_mem_type` - Memory Type attribute file

  This attribute file will display what type of memory is currently on this csrow. Normally, either buffered or unbuffered memory. Examples:

  - Registered-DDR
  - Unbuffered-DDR

`csrow`X directories

When CONFIG_EDAC_LEGACY_SYSFS is enabled, sysfs will contain the `csrow`X directories. As this API doesn't work properly for Rambus, FB-DIMMs and modern Intel Memory Controllers, this is being deprecated in favor of `dimm`X directories.

In the `csrow`X directories are EDAC control and attribute files for this X instance of csrow:

- `ue_count` - Total Uncorrectable Errors count attribute file

  This attribute file displays the total count of uncorrectable errors that have occurred on this csrow. If panic_on_ue is set this counter will not have a chance to increment, since EDAC will panic the system.

- `ce_count` - Total Correctable Errors count attribute file

  This attribute file displays the total count of correctable errors that have occurred on this csrow. This count is very important to examine. CEs provide early indications that a DIMM is beginning to fail. This count field should be monitored for non-zero values and report such information to the system administrator.

- `size_mb` - Total memory managed by this csrow attribute file

This attribute file displays, in count of megabytes, the memory that this csrow contains.

- `mem_type` - Memory Type attribute file

  This attribute file will display what type of memory is currently on this csrow. Normally, either buffered or unbuffered memory. Examples:

  - Registered-DDR
  - Unbuffered-DDR

- `edac_mode` - EDAC Mode of operation attribute file

  This attribute file will display what type of Error detection and correction is being utilized.

- `dev_type` - Device type attribute file

  This attribute file will display what type of DRAM device is being utilized on this DIMM. Examples:

  - x1
  - x2
  - x4
  - x8

- `ch0_ce_count` - Channel 0 CE Count attribute file

  This attribute file will display the count of CEs on this DIMM located in channel 0.

- `ch0_ue_count` - Channel 0 UE Count attribute file

  This attribute file will display the count of UEs on this DIMM located in channel 0.

- `ch0_dimm_label` - Channel 0 DIMM Label control file

  This control file allows this DIMM to have a label assigned to it. With this label in the module, when errors occur the output can provide the DIMM label in the system log. This becomes vital for panic events to isolate the cause of the UE event.

  DIMM Labels must be assigned after booting, with information that correctly identifies the physical slot with its silk screen label. This information is currently very motherboard specific and determination of this information must occur in userland at this time.

- `ch1_ce_count` - Channel 1 CE Count attribute file

  This attribute file will display the count of CEs on this DIMM located in channel 1.

- `ch1_ue_count` - Channel 1 UE Count attribute file

  This attribute file will display the count of UEs on this DIMM located in channel 0.

- `ch1_dimm_label` - Channel 1 DIMM Label control file

  This control file allows this DIMM to have a label assigned to it. With this label in the module, when errors occur the output can provide the DIMM label in the system log. This becomes vital for panic events to isolate the cause of the UE event.

  DIMM Labels must be assigned after booting, with information that correctly identifies the physical slot with its silk screen label. This information is currently very motherboard specific and determination of this information must occur in userland at this time.

System Logging

If logging for UEs and CEs is enabled, then system logs will contain information indicating that errors have been detected:

```
EDAC MC0: CE page 0x283, offset 0xce0, grain 8, syndrome 0x6ec3, row 0, channel 1 "DIMM_B1":␣
 ↪amd76x_edac
EDAC MC0: CE page 0x1e5, offset 0xfb0, grain 8, syndrome 0xb741, row 0, channel 1 "DIMM_B1":␣
 ↪amd76x_edac
```

The structure of the message is:

Content	Example
The memory controller	MC0
Error type	CE
Memory page	0x283
Offset in the page	0xce0
The byte granularity or resolution of the error	grain 8
The error syndrome	0xb741
Memory row	row 0
Memory channel	channel 1
DIMM label, if set prior	DIMM B1
And then an optional, driver-specific message that may have additional information.	

Both UEs and CEs with no info will lack all but memory controller, error type, a notice of “no info” and then an optional, driver-specific error message.

PCI Bus Parity Detection

On Header Type 00 devices, the primary status is looked at for any parity error regardless of whether parity is enabled on the device or not. (The spec indicates parity is generated in some cases). On Header Type 01 bridges, the secondary status register is also looked at to see if parity occurred on the bus on the other side of the bridge.

Sysfs configuration

Under `/sys/devices/system/edac/pci` are control and attribute files as follows:

- `check_pci_parity` - Enable/Disable PCI Parity checking control file

  This control file enables or disables the PCI Bus Parity scanning operation. Writing a 1 to this file enables the scanning. Writing a 0 to this file disables the scanning.

  Enable:

  ```
  echo "1" >/sys/devices/system/edac/pci/check_pci_parity
  ```

  Disable:

  ```
  echo "0" >/sys/devices/system/edac/pci/check_pci_parity
  ```

- `pci_parity_count` - Parity Count

  This attribute file will display the number of parity errors that have been detected.

Module parameters

- `edac_mc_panic_on_ue` - Panic on UE control file

  An uncorrectable error will cause a machine panic. This is usually desirable. It is a bad idea to continue when an uncorrectable error occurs - it is indeterminate what was uncorrected

and the operating system context might be so mangled that continuing will lead to further corruption. If the kernel has MCE configured, then EDAC will never notice the UE.

LOAD TIME:

```
module/kernel parameter: edac_mc_panic_on_ue=[0|1]
```

RUN TIME:

```
echo "1" > /sys/module/edac_core/parameters/edac_mc_panic_on_ue
```

- `edac_mc_log_ue` - Log UE control file

  Generate kernel messages describing uncorrectable errors. These errors are reported through the system message log system. UE statistics will be accumulated even when UE logging is disabled.

  LOAD TIME:

  ```
  module/kernel parameter: edac_mc_log_ue=[0|1]
  ```

  RUN TIME:

  ```
  echo "1" > /sys/module/edac_core/parameters/edac_mc_log_ue
  ```

- `edac_mc_log_ce` - Log CE control file

  Generate kernel messages describing correctable errors. These errors are reported through the system message log system. CE statistics will be accumulated even when CE logging is disabled.

  LOAD TIME:

  ```
  module/kernel parameter: edac_mc_log_ce=[0|1]
  ```

  RUN TIME:

  ```
  echo "1" > /sys/module/edac_core/parameters/edac_mc_log_ce
  ```

- `edac_mc_poll_msec` - Polling period control file

  The time period, in milliseconds, for polling for error information. Too small a value wastes resources. Too large a value might delay necessary handling of errors and might loose valuable information for locating the error. 1000 milliseconds (once each second) is the current default. Systems which require all the bandwidth they can get, may increase this.

  LOAD TIME:

  ```
  module/kernel parameter: edac_mc_poll_msec=[0|1]
  ```

  RUN TIME:

  ```
  echo "1000" > /sys/module/edac_core/parameters/edac_mc_poll_msec
  ```

- `panic_on_pci_parity` - Panic on PCI PARITY Error

  This control file enables or disables panicking when a parity error has been detected.

  module/kernel parameter:

  ```
  edac_panic_on_pci_pe=[0|1]
  ```

  Enable:

```
echo "1" > /sys/module/edac_core/parameters/edac_panic_on_pci_pe
```

Disable:

```
echo "0" > /sys/module/edac_core/parameters/edac_panic_on_pci_pe
```

EDAC device type

In the header file, edac_pci.h, there is a series of edac_device structures and APIs for the EDAC_DEVICE.

User space access to an edac_device is through the sysfs interface.

At the location `/sys/devices/system/edac` (sysfs) new edac_device devices will appear.

There is a three level tree beneath the above edac directory. For example, the `test_device_edac` device (found at the http://bluesmoke.sourceforget.net website) installs itself as:

```
/sys/devices/system/edac/test-instance
```

in this directory are various controls, a symlink and one or more `instance` directories.

The standard default controls are:

log_ce	boolean to log CE events
log_ue	boolean to log UE events
panic_on_ue	boolean to `panic` the system if an UE is encountered (default off, can be set true via startup script)
poll_msec	time period between POLL cycles for events

The test_device_edac device adds at least one of its own custom control:

test_bits	which in the current test driver does nothing but show how it is installed. A ported driver can add one or more such controls and/or attributes for specific uses. One out-of-tree driver uses controls here to allow for ERROR INJECTION operations to hardware injection registers

The symlink points to the 'struct dev' that is registered for this edac_device.

Instances

One or more instance directories are present. For the `test_device_edac` case:

```
test-instance0
```

In this directory there are two default counter attributes, which are totals of counter in deeper subdirectories.

ce_count	total of CE events of subdirectories
ue_count	total of UE events of subdirectories

Blocks

At the lowest directory level is the `block` directory. There can be 0, 1 or more blocks specified in each instance:

```
test-block0
```

In this directory the default attributes are:

ce_count	which is counter of CE events for this `block` of hardware being monitored
ue_count	which is counter of UE events for this `block` of hardware being monitored

The `test_device_edac` device adds 4 attributes and 1 control:

test-block-bits-0	for every POLL cycle this counter is incremented
test-block-bits-1	every 10 cycles, this counter is bumped once, and test-block-bits-0 is set to 0
test-block-bits-2	every 100 cycles, this counter is bumped once, and test-block-bits-1 is set to 0
test-block-bits-3	every 1000 cycles, this counter is bumped once, and test-block-bits-2 is set to 0

reset-counters	writing ANY thing to this control will reset all the above counters.

Use of the `test_device_edac` driver should enable any others to create their own unique drivers for their hardware systems.

The `test_device_edac` sample driver is located at the http://bluesmoke.sourceforge.net project site for EDAC.

Usage of EDAC APIs on Nehalem and newer Intel CPUs

On older Intel architectures, the memory controller was part of the North Bridge chipset. Nehalem, Sandy Bridge, Ivy Bridge, Haswell, Sky Lake and newer Intel architectures integrated an enhanced version of the memory controller (MC) inside the CPUs.

This chapter will cover the differences of the enhanced memory controllers found on newer Intel CPUs, such as `i7core_edac`, `sb_edac` and `sbx_edac` drivers.

Note:

The Xeon E7 processor families use a separate chip for the memory controller, called Intel Scalable Memory Buffer. This section doesn't apply for such families.

1. There is one Memory Controller per Quick Patch Interconnect (QPI). At the driver, the term "socket" means one QPI. This is associated with a physical CPU socket.

   Each MC have 3 physical read channels, 3 physical write channels and 3 logic channels. The driver currently sees it as just 3 channels. Each channel can have up to 3 DIMMs.

   The minimum known unity is DIMMs. There are no information about csrows. As EDAC API maps the minimum unity is csrows, the driver sequentially maps channel/DIMM into different csrows.

   For example, supposing the following layout:

```
Ch0 phy rd0, wr0 (0x063f4031): 2 ranks, UDIMMs
  dimm 0 1024 Mb offset: 0, bank: 8, rank: 1, row: 0x4000, col: 0x400
  dimm 1 1024 Mb offset: 4, bank: 8, rank: 1, row: 0x4000, col: 0x400
Ch1 phy rd1, wr1 (0x063f4031): 2 ranks, UDIMMs
  dimm 0 1024 Mb offset: 0, bank: 8, rank: 1, row: 0x4000, col: 0x400
Ch2 phy rd3, wr3 (0x063f4031): 2 ranks, UDIMMs
  dimm 0 1024 Mb offset: 0, bank: 8, rank: 1, row: 0x4000, col: 0x400
```

   The driver will map it as:

```
csrow0: channel 0, dimm0
csrow1: channel 0, dimm1
csrow2: channel 1, dimm0
csrow3: channel 2, dimm0
```

   exports one DIMM per csrow.

   Each QPI is exported as a different memory controller.

2. The MC has the ability to inject errors to test drivers. The drivers implement this functionality via some error injection nodes:

   For injecting a memory error, there are some sysfs nodes, under /sys/devices/system/edac/mc/mc?/:

   - **inject_addrmatch/*:** Controls the error injection mask register. It is possible to specify several characteristics of the address to match an error code:

     ```
     dimm = the affected dimm. Numbers are relative to a channel;
     rank = the memory rank;
     channel = the channel that will generate an error;
     bank = the affected bank;
     page = the page address;
     column (or col) = the address column.
     ```

     each of the above values can be set to “any” to match any valid value.

     At driver init, all values are set to any.

     For example, to generate an error at rank 1 of dimm 2, for any channel, any bank, any page, any column:

     ```
             echo 2 >/sys/devices/system/edac/mc/mc0/inject_addrmatch/dimm
             echo 1 >/sys/devices/system/edac/mc/mc0/inject_addrmatch/rank

     To return to the default behaviour of matching any, you can do::

             echo any >/sys/devices/system/edac/mc/mc0/inject_addrmatch/dimm
             echo any >/sys/devices/system/edac/mc/mc0/inject_addrmatch/rank
     ```

   - **inject_eccmask:** specifies what bits will have troubles,
   - **inject_section:** specifies what ECC cache section will get the error:

     ```
     3 for both
     2 for the highest
     1 for the lowest
     ```

   - **inject_type:** specifies the type of error, being a combination of the following bits:

     ```
     bit 0 - repeat
     bit 1 - ecc
     bit 2 - parity
     ```

   - **inject_enable:** starts the error generation when something different than 0 is written.

   All inject vars can be read. root permission is needed for write.

   Datasheet states that the error will only be generated after a write on an address that matches inject_addrmatch. It seems, however, that reading will also produce an error.

   For example, the following code will generate an error for any write access at socket 0, on any DIMM/address on channel 2:

   ```
   echo 2 >/sys/devices/system/edac/mc/mc0/inject_addrmatch/channel
   echo 2 >/sys/devices/system/edac/mc/mc0/inject_type
   echo 64 >/sys/devices/system/edac/mc/mc0/inject_eccmask
   echo 3 >/sys/devices/system/edac/mc/mc0/inject_section
   echo 1 >/sys/devices/system/edac/mc/mc0/inject_enable
   dd if=/dev/mem of=/dev/null seek=16k bs=4k count=1 >& /dev/null
   ```

   For socket 1, it is needed to replace “mc0” by “mc1” at the above commands.

   The generated error message will look like:

```
EDAC MC0: UE row 0, channel-a= 0 channel-b= 0 labels "-": NON_FATAL (addr = 0x0075b980,␣
↪socket=0, Dimm=0, Channel=2, syndrome=0x00000040, count=1, Err=8c0000400001009f:
↪4000080482 (read error: read ECC error))
```

3. Corrected Error memory register counters

   Those newer MCs have some registers to count memory errors. The driver uses those registers to report Corrected Errors on devices with Registered DIMMs.

   However, those counters don't work with Unregistered DIMM. As the chipset offers some counters that also work with UDIMMs (but with a worse level of granularity than the default ones), the driver exposes those registers for UDIMM memories.

   They can be read by looking at the contents of `all_channel_counts/`:

```
$ for i in /sys/devices/system/edac/mc/mc0/all_channel_counts/*; do echo $i; cat $i; done
   /sys/devices/system/edac/mc/mc0/all_channel_counts/udimm0
   0
   /sys/devices/system/edac/mc/mc0/all_channel_counts/udimm1
   0
   /sys/devices/system/edac/mc/mc0/all_channel_counts/udimm2
   0
```

   What happens here is that errors on different csrows, but at the same dimm number will increment the same counter. So, in this memory mapping:

```
csrow0: channel 0, dimm0
csrow1: channel 0, dimm1
csrow2: channel 1, dimm0
csrow3: channel 2, dimm0
```

   The hardware will increment udimm0 for an error at the first dimm at either csrow0, csrow2 or csrow3;

   The hardware will increment udimm1 for an error at the second dimm at either csrow0, csrow2 or csrow3;

   The hardware will increment udimm2 for an error at the third dimm at either csrow0, csrow2 or csrow3;

4. Standard error counters

   The standard error counters are generated when an mcelog error is received by the driver. Since, with UDIMM, this is counted by software, it is possible that some errors could be lost. With RDIMM's, they display the contents of the registers

Reference documents used on amd64_edac

amd64_edac module is based on the following documents (available from http://support.amd.com/en-us/search/tech-docs):

1. **Title** BIOS and Kernel Developer's Guide for AMD Athlon 64 and AMD Opteron Processors

   AMD publication # 26094

   Revision 3.26

   Link http://support.amd.com/TechDocs/26094.PDF

2. **Title** BIOS and Kernel Developer's Guide for AMD NPT Family 0Fh Processors

   AMD publication # 32559

   Revision 3.00

   Issue Date May 2006

Link http://support.amd.com/TechDocs/32559.pdf

3. **Title** BIOS and Kernel Developer's Guide (BKDG) For AMD Family 10h Processors

   AMD publication # 31116

   Revision 3.00

   Issue Date September 07, 2007

   Link http://support.amd.com/TechDocs/31116.pdf

4. **Title** BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 30h-3Fh Processors

   AMD publication # 49125

   Revision 3.06

   Issue Date 2/12/2015 (latest release)

   Link http://support.amd.com/TechDocs/49125_15h_Models_30h-3Fh_BKDG.pdf

5. **Title** BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 60h-6Fh Processors

   AMD publication # 50742

   Revision 3.01

   Issue Date 7/23/2015 (latest release)

   Link http://support.amd.com/TechDocs/50742_15h_Models_60h-6Fh_BKDG.pdf

6. **Title** BIOS and Kernel Developer's Guide (BKDG) for AMD Family 16h Models 00h-0Fh Processors

   AMD publication # 48751

   Revision 3.03

   Issue Date 2/23/2015 (latest release)

   Link http://support.amd.com/TechDocs/48751_16h_bkdg.pdf

Credits

- Written by Doug Thompson <dougthompson@xmission.com>
  - 7 Dec 2005
  - 17 Jul 2007 Updated
- © Mauro Carvalho Chehab
  - 05 Aug 2009 Nehalem interface
  - 26 Oct 2016 Converted to ReST and cleanups at the Nehalem section
- EDAC authors/maintainers:
  - Doug Thompson, Dave Jiang, Dave Peterson et al,
  - Mauro Carvalho Chehab
  - Borislav Petkov
  - original author: Thayne Harbaugh

CHAPTER

TWENTYSIX

POWER MANAGEMENT

Power Management Strategies

The Linux kernel supports two major high-level power management strategies.

One of them is based on using global low-power states of the whole system in which user space code cannot be executed and the overall system activity is significantly reduced, referred to as *sleep states* . The kernel puts the system into one of these states when requested by user space and the system stays in it until a special signal is received from one of designated devices, triggering a transition to the `working state` in which user space code can run. Because sleep states are global and the whole system is affected by the state changes, this strategy is referred to as the *system-wide power management* .

The other strategy, referred to as the *working-state power management* , is based on adjusting the power states of individual hardware components of the system, as needed, in the working state. In consequence, if this strategy is in use, the working state of the system usually does not correspond to any particular physical configuration of it, but can be treated as a metastate covering a range of different power states of the system in which the individual components of it can be either `active` (in use) or `inactive` (idle). If they are active, they have to be in power states allowing them to process data and to be accessed by software. In turn, if they are inactive, ideally, they should be in low-power states in which they may not be accessible.

If all of the system components are active, the system as a whole is regarded as “runtime active” and that situation typically corresponds to the maximum power draw (or maximum energy usage) of it. If all of them are inactive, the system as a whole is regarded as “runtime idle” which may be very close to a sleep state from the physical system configuration and power draw perspective, but then it takes much less time and effort to start executing user space code than for the same system in a sleep state. However, transitions from sleep states back to the working state can only be started by a limited set of devices, so typically the system can spend much more time in a sleep state than it can be runtime idle in one go. For this reason, systems usually use less energy in sleep states than when they are runtime idle most of the time.

Moreover, the two power management strategies address different usage scenarios. Namely, if the user indicates that the system will not be in use going forward, for example by closing its lid (if the system is a laptop), it probably should go into a sleep state at that point. On the other hand, if the user simply goes away from the laptop keyboard, it probably should stay in the working state and use the working-state power management in case it becomes idle, because the user may come back to it at any time and then may want the system to be immediately accessible.

System-Wide Power Management

System Sleep States

Sleep states are global low-power states of the entire system in which user space code cannot be executed and the overall system activity is significantly reduced.

Sleep States That Can Be Supported

Depending on its configuration and the capabilities of the platform it runs on, the Linux kernel can support up to four system sleep states, includig hibernation and up to three variants of system suspend. The sleep states that can be supported by the kernel are listed below.

Suspend-to-Idle

This is a generic, pure software, light-weight variant of system suspend (also referred to as S2I or S2Idle). It allows more energy to be saved relative to runtime idle by freezing user space, suspending the timekeeping and putting all I/O devices into low-power states (possibly lower-power than available in the working state), such that the processors can spend time in their deepest idle states while the system is suspended.

The system is woken up from this state by in-band interrupts, so theoretically any devices that can cause interrupts to be generated in the working state can also be set up as wakeup devices for S2Idle.

This state can be used on platforms without support for *standby* or *suspend-to-RAM* , or it can be used in addition to any of the deeper system suspend variants to provide reduced resume latency. It is always supported if the `CONFIG_SUSPEND` kernel configuration option is set.

Standby

This state, if supported, offers moderate, but real, energy savings, while providing a relatively straightforward transition back to the working state. No operating state is lost (the system core logic retains power), so the system can go back to where it left off easily enough.

In addition to freezing user space, suspending the timekeeping and putting all I/O devices into low-power states, which is done for *suspend-to-idle* too, nonboot CPUs are taken offline and all low-level system functions are suspended during transitions into this state. For this reason, it should allow more energy to be saved relative to *suspend-to-idle* , but the resume latency will generally be greater than for that state.

The set of devices that can wake up the system from this state usually is reduced relative to *suspend-to-idle* and it may be necessary to rely on the platform for setting up the wakeup functionality as appropriate.

This state is supported if the `CONFIG_SUSPEND` kernel configuration option is set and the support for it is registered by the platform with the core system suspend subsystem. On ACPI-based systems this state is mapped to the S1 system state defined by ACPI.

Suspend-to-RAM

This state (also referred to as STR or S2RAM), if supported, offers significant energy savings as everything in the system is put into a low-power state, except for memory, which should be placed into the self-refresh mode to retain its contents. All of the steps carried out when entering *standby* are also carried out during transitions to S2RAM. Additional operations may take place depending on the platform capabilities. In particular, on ACPI-based systems the kernel passes control to the platform firmware (BIOS) as the last step during S2RAM transitions and that usually results in powering down some more low-level components that are not directly controlled by the kernel.

The state of devices and CPUs is saved and held in memory. All devices are suspended and put into low-power states. In many cases, all peripheral buses lose power when entering S2RAM, so devices must be able to handle the transition back to the "on" state.

On ACPI-based systems S2RAM requires some minimal boot-strapping code in the platform firmware to resume the system from it. This may be the case on other platforms too.

The set of devices that can wake up the system from S2RAM usually is reduced relative to *suspend-to-idle* and *standby* and it may be necessary to rely on the platform for setting up the wakeup functionality as appropriate.

S2RAM is supported if the `CONFIG_SUSPEND` kernel configuration option is set and the support for it is registered by the platform with the core system suspend subsystem. On ACPI-based systems it is mapped to the S3 system state defined by ACPI.

Hibernation

This state (also referred to as Suspend-to-Disk or STD) offers the greatest energy savings and can be used even in the absence of low-level platform support for system suspend. However, it requires some low-level code for resuming the system to be present for the underlying CPU architecture.

Hibernation is significantly different from any of the system suspend variants. It takes three system state changes to put it into hibernation and two system state changes to resume it.

First, when hibernation is triggered, the kernel stops all system activity and creates a snapshot image of memory to be written into persistent storage. Next, the system goes into a state in which the snapshot image can be saved, the image is written out and finally the system goes into the target low-power state in which power is cut from almost all of its hardware components, including memory, except for a limited set of wakeup devices.

Once the snapshot image has been written out, the system may either enter a special low-power state (like ACPI S4), or it may simply power down itself. Powering down means minimum power draw and it allows this mechanism to work on any system. However, entering a special low-power state may allow additional means of system wakeup to be used (e.g. pressing a key on the keyboard or opening a laptop lid).

After wakeup, control goes to the platform firmware that runs a boot loader which boots a fresh instance of the kernel (control may also go directly to the boot loader, depending on the system configuration, but anyway it causes a fresh instance of the kernel to be booted). That new instance of the kernel (referred to as the `restore kernel`) looks for a hibernation image in persistent storage and if one is found, it is loaded into memory. Next, all activity in the system is stopped and the restore kernel overwrites itself with the image contents and jumps into a special trampoline area in the original kernel stored in the image (referred to as the `image kernel`), which is where the special architecture-specific low-level code is needed. Finally, the image kernel restores the system to the pre-hibernation state and allows user space to run again.

Hibernation is supported if the `CONFIG_HIBERNATION` kernel configuration option is set. However, this option can only be set if support for the given CPU architecture includes the low-level code for system resume.

Basic `sysfs` Interfaces for System Suspend and Hibernation

The following files located in the `/sys/power/` directory can be used by user space for sleep states control.

`state` This file contains a list of strings representing sleep states supported by the kernel. Writing one of these strings into it causes the kernel to start a transition of the system into the sleep state represented by that string.

In particular, the strings "disk", "freeze" and "standby" represent the *hibernation*, *suspend-to-idle* and *standby* sleep states, respectively. The string "mem" is interpreted in accordance with the contents of the `mem_sleep` file described below.

If the kernel does not support any system sleep states, this file is not present.

mem_sleep This file contains a list of strings representing supported system suspend variants and allows user space to select the variant to be associated with the "mem" string in the `state` file described above.

The strings that may be present in this file are "s2idle", "shallow" and "deep". The string "s2idle" always represents *suspend-to-idle* and, by convention, "shallow" and "deep" represent *standby* and *suspend-to-RAM*, respectively.

Writing one of the listed strings into this file causes the system suspend variant represented by it to be associated with the "mem" string in the `state` file. The string representing the suspend variant currently associated with the "mem" string in the `state` file is listed in square brackets.

If the kernel does not support system suspend, this file is not present.

disk This file contains a list of strings representing different operations that can be carried out after the hibernation image has been saved. The possible options are as follows:

platform Put the system into a special low-power state (e.g. ACPI S4) to make additional wakeup options available and possibly allow the platform firmware to take a simplified initialization path after wakeup.

shutdown Power off the system.

reboot Reboot the system (useful for diagnostics mostly).

suspend Hybrid system suspend. Put the system into the suspend sleep state selected through the `mem_sleep` file described above. If the system is successfully woken up from that state, discard the hibernation image and continue. Otherwise, use the image to restore the previous state of the system.

test_resume Diagnostic operation. Load the image as though the system had just woken up from hibernation and the currently running kernel instance was a restore kernel and follow up with full system resume.

Writing one of the listed strings into this file causes the option represented by it to be selected.

The currently selected option is shown in square brackets which means that the operation represented by it will be carried out after creating and saving the image next time hibernation is triggered by writing `disk` to `/sys/power/state`.

If the kernel does not support hibernation, this file is not present.

According to the above, there are two ways to make the system go into the *suspend-to-idle* state. The first one is to write "freeze" directly to `/sys/power/state`. The second one is to write "s2idle" to `/sys/power/mem_sleep` and then to write "mem" to `/sys/power/state`. Likewise, there are two ways to make the system go into the *standby* state (the strings to write to the control files in that case are "standby" or "shallow" and "mem", respectively) if that state is supported by the platform. However, there is only one way to make the system go into the *suspend-to-RAM* state (write "deep" into `/sys/power/mem_sleep` and "mem" into `/sys/power/state`).

The default suspend variant (ie. the one to be used without writing anything into `/sys/power/mem_sleep`) is either "deep" (on the majority of systems supporting *suspend-to-RAM*) or "s2idle", but it can be overridden by the value of the "mem_sleep_default" parameter in the kernel command line. On some ACPI-based systems, depending on the information in the ACPI tables, the default may be "s2idle" even if *suspend-to-RAM* is supported.

Working-State Power Management

CPU Performance Scaling

The Concept of CPU Performance Scaling

The majority of modern processors are capable of operating in a number of different clock frequency and voltage configurations, often referred to as Operating Performance Points or P-states (in ACPI terminology). As a rule, the higher the clock frequency and the higher the voltage, the more instructions can be retired by the CPU over a unit of time, but also the higher the clock frequency and the higher the voltage, the more energy is consumed over a unit of time (or the more power is drawn) by the CPU in the given P-state. Therefore there is a natural tradeoff between the CPU capacity (the number of instructions that can be executed over a unit of time) and the power drawn by the CPU.

In some situations it is desirable or even necessary to run the program as fast as possible and then there is no reason to use any P-states different from the highest one (i.e. the highest-performance frequency/voltage configuration available). In some other cases, however, it may not be necessary to execute instructions so quickly and maintaining the highest available CPU capacity for a relatively long time without utilizing it entirely may be regarded as wasteful. It also may not be physically possible to maintain maximum CPU capacity for too long for thermal or power supply capacity reasons or similar. To cover those cases, there are hardware interfaces allowing CPUs to be switched between different frequency/voltage configurations or (in the ACPI terminology) to be put into different P-states.

Typically, they are used along with algorithms to estimate the required CPU capacity, so as to decide which P-states to put the CPUs into. Of course, since the utilization of the system generally changes over time, that has to be done repeatedly on a regular basis. The activity by which this happens is referred to as CPU performance scaling or CPU frequency scaling (because it involves adjusting the CPU clock frequency).

CPU Performance Scaling in Linux

The Linux kernel supports CPU performance scaling by means of the CPUFreq (CPU Frequency scaling) subsystem that consists of three layers of code: the core, scaling governors and scaling drivers.

The CPUFreq core provides the common code infrastructure and user space interfaces for all platforms that support CPU performance scaling. It defines the basic framework in which the other components operate.

Scaling governors implement algorithms to estimate the required CPU capacity. As a rule, each governor implements one, possibly parametrized, scaling algorithm.

Scaling drivers talk to the hardware. They provide scaling governors with information on the available P-states (or P-state ranges in some cases) and access platform-specific hardware interfaces to change CPU P-states as requested by scaling governors.

In principle, all available scaling governors can be used with every scaling driver. That design is based on the observation that the information used by performance scaling algorithms for P-state selection can be represented in a platform-independent form in the majority of cases, so it should be possible to use the same performance scaling algorithm implemented in exactly the same way regardless of which scaling driver is used. Consequently, the same set of scaling governors should be suitable for every supported platform.

However, that observation may not hold for performance scaling algorithms based on information provided by the hardware itself, for example through feedback registers, as that information is typically specific to the hardware interface it comes from and may not be easily represented in an abstract, platform-independent way. For this reason, CPUFreq allows scaling drivers to bypass the governor layer and implement their own performance scaling algorithms. That is done by the *intel_pstate* scaling driver.

CPUFreq Policy Objects

In some cases the hardware interface for P-state control is shared by multiple CPUs. That is, for example, the same register (or set of registers) is used to control the P-state of multiple CPUs at the same time and writing to it affects all of those CPUs simultaneously.

Sets of CPUs sharing hardware P-state control interfaces are represented by CPUFreq as `struct cpufreq_policy` objects. For consistency, `struct cpufreq_policy` is also used when there is only one CPU in the given set.

The CPUFreq core maintains a pointer to a `struct cpufreq_policy` object for every CPU in the system, including CPUs that are currently offline. If multiple CPUs share the same hardware P-state control interface, all of the pointers corresponding to them point to the same `struct cpufreq_policy` object.

CPUFreq uses `struct cpufreq_policy` as its basic data type and the design of its user space interface is based on the policy concept.

CPU Initialization

First of all, a scaling driver has to be registered for CPUFreq to work. It is only possible to register one scaling driver at a time, so the scaling driver is expected to be able to handle all CPUs in the system.

The scaling driver may be registered before or after CPU registration. If CPUs are registered earlier, the driver core invokes the CPUFreq core to take a note of all of the already registered CPUs during the registration of the scaling driver. In turn, if any CPUs are registered after the registration of the scaling driver, the CPUFreq core will be invoked to take note of them at their registration time.

In any case, the CPUFreq core is invoked to take note of any logical CPU it has not seen so far as soon as it is ready to handle that CPU. [Note that the logical CPU may be a physical single-core processor, or a single core in a multicore processor, or a hardware thread in a physical processor or processor core. In what follows “CPU” always means “logical CPU” unless explicitly stated otherwise and the word “processor” is used to refer to the physical part possibly including multiple logical CPUs.]

Once invoked, the CPUFreq core checks if the policy pointer is already set for the given CPU and if so, it skips the policy object creation. Otherwise, a new policy object is created and initialized, which involves the creation of a new policy directory in `sysfs`, and the policy pointer corresponding to the given CPU is set to the new policy object’s address in memory.

Next, the scaling driver’s `->init()` callback is invoked with the policy pointer of the new CPU passed to it as the argument. That callback is expected to initialize the performance scaling hardware interface for the given CPU (or, more precisely, for the set of CPUs sharing the hardware interface it belongs to, represented by its policy object) and, if the policy object it has been called for is new, to set parameters of the policy, like the minimum and maximum frequencies supported by the hardware, the table of available frequencies (if the set of supported P-states is not a continuous range), and the mask of CPUs that belong to the same policy (including both online and offline CPUs). That mask is then used by the core to populate the policy pointers for all of the CPUs in it.

The next major initialization step for a new policy object is to attach a scaling governor to it (to begin with, that is the default scaling governor determined by the kernel configuration, but it may be changed later via `sysfs`). First, a pointer to the new policy object is passed to the governor’s `->init()` callback which is expected to initialize all of the data structures necessary to handle the given policy and, possibly, to add a governor `sysfs` interface to it. Next, the governor is started by invoking its `->start()` callback.

That callback it expected to register per-CPU utilization update callbacks for all of the online CPUs belonging to the given policy with the CPU scheduler. The utilization update callbacks will be invoked by the CPU scheduler on important events, like task enqueue and dequeue, on every iteration of the scheduler tick or generally whenever the CPU utilization may change (from the scheduler’s perspective). They are expected to carry out computations needed to determine the P-state to use for the given policy going forward and to invoke the scaling driver to make changes to the hardware in accordance with the P-state selection. The scaling driver may be invoked directly from scheduler context or asynchronously, via a kernel thread or workqueue, depending on the configuration and capabilities of the scaling driver and the governor.

Similar steps are taken for policy objects that are not new, but were "inactive" previously, meaning that all of the CPUs belonging to them were offline. The only practical difference in that case is that the CPUFreq core will attempt to use the scaling governor previously used with the policy that became "inactive" (and is re-initialized now) instead of the default governor.

In turn, if a previously offline CPU is being brought back online, but some other CPUs sharing the policy object with it are online already, there is no need to re-initialize the policy object at all. In that case, it only is necessary to restart the scaling governor so that it can take the new online CPU into account. That is achieved by invoking the governor's `->stop` and `->start()` callbacks, in this order, for the entire policy.

As mentioned before, the *intel_pstate* scaling driver bypasses the scaling governor layer of CPUFreq and provides its own P-state selection algorithms. Consequently, if *intel_pstate* is used, scaling governors are not attached to new policy objects. Instead, the driver's `->setpolicy()` callback is invoked to register per-CPU utilization update callbacks for each policy. These callbacks are invoked by the CPU scheduler in the same way as for scaling governors, but in the *intel_pstate* case they both determine the P-state to use and change the hardware configuration accordingly in one go from scheduler context.

The policy objects created during CPU initialization and other data structures associated with them are torn down when the scaling driver is unregistered (which happens when the kernel module containing it is unloaded, for example) or when the last CPU belonging to the given policy in unregistered.

Policy Interface in sysfs

During the initialization of the kernel, the CPUFreq core creates a `sysfs` directory (kobject) called `cpufreq` under `/sys/devices/system/cpu/`.

That directory contains a `policyX` subdirectory (where X represents an integer number) for every policy object maintained by the CPUFreq core. Each `policyX` directory is pointed to by `cpufreq` symbolic links under `/sys/devices/system/cpu/cpuY/` (where Y represents an integer that may be different from the one represented by X) for all of the CPUs associated with (or belonging to) the given policy. The `policyX` directories in `/sys/devices/system/cpu/cpufreq` each contain policy-specific attributes (files) to control CPUFreq behavior for the corresponding policy objects (that is, for all of the CPUs associated with them).

Some of those attributes are generic. They are created by the CPUFreq core and their behavior generally does not depend on what scaling driver is in use and what scaling governor is attached to the given policy. Some scaling drivers also add driver-specific attributes to the policy directories in `sysfs` to control policy-specific aspects of driver behavior.

The generic attributes under `/sys/devices/system/cpu/cpufreq/policyX/` are the following:

`affected_cpus` List of online CPUs belonging to this policy (i.e. sharing the hardware performance scaling interface represented by the `policyX` policy object).

`bios_limit` If the platform firmware (BIOS) tells the OS to apply an upper limit to CPU frequencies, that limit will be reported through this attribute (if present).

The existence of the limit may be a result of some (often unintentional) BIOS settings, restrictions coming from a service processor or another BIOS/HW-based mechanisms.

This does not cover ACPI thermal limitations which can be discovered through a generic thermal driver.

This attribute is not present if the scaling driver in use does not support it.

`cpuinfo_cur_freq` Current frequency of the CPUs belonging to this policy as obtained from the hardware (in KHz).

This is expected to be the frequency the hardware actually runs at. If that frequency cannot be determined, this attribute should not be present.

`cpuinfo_max_freq` Maximum possible operating frequency the CPUs belonging to this policy can run at (in kHz).

`cpuinfo_min_freq` Minimum possible operating frequency the CPUs belonging to this policy can run at (in kHz).

cpuinfo_transition_latency The time it takes to switch the CPUs belonging to this policy from one P-state to another, in nanoseconds.

If unknown or if known to be so high that the scaling driver does not work with the *ondemand* governor, -1 (CPUFREQ_ETERNAL) will be returned by reads from this attribute.

related_cpus List of all (online and offline) CPUs belonging to this policy.

scaling_available_governors List of CPUFreq scaling governors present in the kernel that can be attached to this policy or (if the *intel_pstate* scaling driver is in use) list of scaling algorithms provided by the driver that can be applied to this policy.

[Note that some governors are modular and it may be necessary to load a kernel module for the governor held by it to become available and be listed by this attribute.]

scaling_cur_freq Current frequency of all of the CPUs belonging to this policy (in kHz).

In the majority of cases, this is the frequency of the last P-state requested by the scaling driver from the hardware using the scaling interface provided by it, which may or may not reflect the frequency the CPU is actually running at (due to hardware design and other limitations).

Some architectures (e.g. x86) may attempt to provide information more precisely reflecting the current CPU frequency through this attribute, but that still may not be the exact current CPU frequency as seen by the hardware at the moment.

scaling_driver The scaling driver currently in use.

scaling_governor The scaling governor currently attached to this policy or (if the *intel_pstate* scaling driver is in use) the scaling algorithm provided by the driver that is currently applied to this policy.

This attribute is read-write and writing to it will cause a new scaling governor to be attached to this policy or a new scaling algorithm provided by the scaling driver to be applied to it (in the *intel_pstate* case), as indicated by the string written to this attribute (which must be one of the names listed by the scaling_available_governors attribute described above).

scaling_max_freq Maximum frequency the CPUs belonging to this policy are allowed to be running at (in kHz).

This attribute is read-write and writing a string representing an integer to it will cause a new limit to be set (it must not be lower than the value of the scaling_min_freq attribute).

scaling_min_freq Minimum frequency the CPUs belonging to this policy are allowed to be running at (in kHz).

This attribute is read-write and writing a string representing a non-negative integer to it will cause a new limit to be set (it must not be higher than the value of the scaling_max_freq attribute).

scaling_setspeed This attribute is functional only if the *userspace* scaling governor is attached to the given policy.

It returns the last frequency requested by the governor (in kHz) or can be written to in order to set a new frequency for the policy.

Generic Scaling Governors

CPUFreq provides generic scaling governors that can be used with all scaling drivers. As stated before, each of them implements a single, possibly parametrized, performance scaling algorithm.

Scaling governors are attached to policy objects and different policy objects can be handled by different scaling governors at the same time (although that may lead to suboptimal results in some cases).

The scaling governor for a given policy object can be changed at any time with the help of the scaling_governor policy attribute in sysfs.

Some governors expose sysfs attributes to control or fine-tune the scaling algorithms implemented by them. Those attributes, referred to as governor tunables, can be either global (system-wide) or per-policy, depending on the scaling driver in use. If the driver requires governor tunables to be per-policy,

they are located in a subdirectory of each policy directory. Otherwise, they are located in a subdirectory under `/sys/devices/system/cpu/cpufreq/`. In either case the name of the subdirectory containing the governor tunables is the name of the governor providing them.

performance

When attached to a policy object, this governor causes the highest frequency, within the `scaling_max_freq` policy limit, to be requested for that policy.

The request is made once at that time the governor for the policy is set to `performance` and whenever the `scaling_max_freq` or `scaling_min_freq` policy limits change after that.

powersave

When attached to a policy object, this governor causes the lowest frequency, within the `scaling_min_freq` policy limit, to be requested for that policy.

The request is made once at that time the governor for the policy is set to `powersave` and whenever the `scaling_max_freq` or `scaling_min_freq` policy limits change after that.

userspace

This governor does not do anything by itself. Instead, it allows user space to set the CPU frequency for the policy it is attached to by writing to the `scaling_setspeed` attribute of that policy.

schedutil

This governor uses CPU utilization data available from the CPU scheduler. It generally is regarded as a part of the CPU scheduler, so it can access the scheduler's internal data structures directly.

It runs entirely in scheduler context, although in some cases it may need to invoke the scaling driver asynchronously when it decides that the CPU frequency should be changed for a given policy (that depends on whether or not the driver is capable of changing the CPU frequency from scheduler context).

The actions of this governor for a particular CPU depend on the scheduling class invoking its utilization update callback for that CPU. If it is invoked by the RT or deadline scheduling classes, the governor will increase the frequency to the allowed maximum (that is, the `scaling_max_freq` policy limit). In turn, if it is invoked by the CFS scheduling class, the governor will use the Per-Entity Load Tracking (PELT) metric for the root control group of the given CPU as the CPU utilization estimate (see the Per-entity load tracking LWN.net article for a description of the PELT mechanism). Then, the new CPU frequency to apply is computed in accordance with the formula

    f = 1.25 * f_0 * util / max

where `util` is the PELT number, `max` is the theoretical maximum of `util`, and `f_0` is either the maximum possible CPU frequency for the given policy (if the PELT number is frequency-invariant), or the current CPU frequency (otherwise).

This governor also employs a mechanism allowing it to temporarily bump up the CPU frequency for tasks that have been waiting on I/O most recently, called "IO-wait boosting". That happens when the `SCHED_CPUFREQ_IOWAIT` flag is passed by the scheduler to the governor callback which causes the frequency to go up to the allowed maximum immediately and then draw back to the value returned by the above formula over time.

This governor exposes only one tunable:

rate_limit_us Minimum time (in microseconds) that has to pass between two consecutive runs of governor computations (default: 1000 times the scaling driver's transition latency).

The purpose of this tunable is to reduce the scheduler context overhead of the governor which might be excessive without it.

This governor generally is regarded as a replacement for the older *ondemand* and *conservative* governors (described below), as it is simpler and more tightly integrated with the CPU scheduler, its overhead in terms of CPU context switches and similar is less significant, and it uses the scheduler's own CPU utilization metric, so in principle its decisions should not contradict the decisions made by the other parts of the scheduler.

ondemand

This governor uses CPU load as a CPU frequency selection metric.

In order to estimate the current CPU load, it measures the time elapsed between consecutive invocations of its worker routine and computes the fraction of that time in which the given CPU was not idle. The ratio of the non-idle (active) time to the total CPU time is taken as an estimate of the load.

If this governor is attached to a policy shared by multiple CPUs, the load is estimated for all of them and the greatest result is taken as the load estimate for the entire policy.

The worker routine of this governor has to run in process context, so it is invoked asynchronously (via a workqueue) and CPU P-states are updated from there if necessary. As a result, the scheduler context overhead from this governor is minimum, but it causes additional CPU context switches to happen relatively often and the CPU P-state updates triggered by it can be relatively irregular. Also, it affects its own CPU load metric by running code that reduces the CPU idle time (even though the CPU idle time is only reduced very slightly by it).

It generally selects CPU frequencies proportional to the estimated load, so that the value of the `cpuinfo_max_freq` policy attribute corresponds to the load of 1 (or 100%), and the value of the `cpuinfo_min_freq` policy attribute corresponds to the load of 0, unless when the load exceeds a (configurable) speedup threshold, in which case it will go straight for the highest frequency it is allowed to use (the `scaling_max_freq` policy limit).

This governor exposes the following tunables:

`sampling_rate` This is how often the governor's worker routine should run, in microseconds.

Typically, it is set to values of the order of 10000 (10 ms). Its default value is equal to the value of `cpuinfo_transition_latency` for each policy this governor is attached to (but since the unit here is greater by 1000, this means that the time represented by `sampling_rate` is 1000 times greater than the transition latency by default).

If this tunable is per-policy, the following shell command sets the time represented by it to be 750 times as high as the transition latency:

```
# echo `$(($(cat cpuinfo_transition_latency) * 750 / 1000)) > ondemand/sampling_rate
```

`up_threshold` If the estimated CPU load is above this value (in percent), the governor will set the frequency to the maximum value allowed for the policy. Otherwise, the selected frequency will be proportional to the estimated CPU load.

`ignore_nice_load` If set to 1 (default 0), it will cause the CPU load estimation code to treat the CPU time spent on executing tasks with "nice" levels greater than 0 as CPU idle time.

This may be useful if there are tasks in the system that should not be taken into account when deciding what frequency to run the CPUs at. Then, to make that happen it is sufficient to increase the "nice" level of those tasks above 0 and set this attribute to 1.

`sampling_down_factor` Temporary multiplier, between 1 (default) and 100 inclusive, to apply to the `sampling_rate` value if the CPU load goes above `up_threshold`.

This causes the next execution of the governor's worker routine (after setting the frequency to the allowed maximum) to be delayed, so the frequency stays at the maximum level for a longer time.

Frequency fluctuations in some bursty workloads may be avoided this way at the cost of additional energy spent on maintaining the maximum CPU capacity.

powersave_bias Reduction factor to apply to the original frequency target of the governor (including the maximum value used when the `up_threshold` value is exceeded by the estimated CPU load) or sensitivity threshold for the AMD frequency sensitivity powersave bias driver (`drivers/cpufreq/amd_freq_sensitivity.c`), between 0 and 1000 inclusive.

If the AMD frequency sensitivity powersave bias driver is not loaded, the effective frequency to apply is given by

f * (1 - `powersave_bias` / 1000)

where f is the governor's original frequency target. The default value of this attribute is 0 in that case.

If the AMD frequency sensitivity powersave bias driver is loaded, the value of this attribute is 400 by default and it is used in a different way.

On Family 16h (and later) AMD processors there is a mechanism to get a measured workload sensitivity, between 0 and 100% inclusive, from the hardware. That value can be used to estimate how the performance of the workload running on a CPU will change in response to frequency changes.

The performance of a workload with the sensitivity of 0 (memory-bound or IO-bound) is not expected to increase at all as a result of increasing the CPU frequency, whereas workloads with the sensitivity of 100% (CPU-bound) are expected to perform much better if the CPU frequency is increased.

If the workload sensitivity is less than the threshold represented by the `powersave_bias` value, the sensitivity powersave bias driver will cause the governor to select a frequency lower than its original target, so as to avoid over-provisioning workloads that will not benefit from running at higher CPU frequencies.

conservative

This governor uses CPU load as a CPU frequency selection metric.

It estimates the CPU load in the same way as the *ondemand* governor described above, but the CPU frequency selection algorithm implemented by it is different.

Namely, it avoids changing the frequency significantly over short time intervals which may not be suitable for systems with limited power supply capacity (e.g. battery-powered). To achieve that, it changes the frequency in relatively small steps, one step at a time, up or down - depending on whether or not a (configurable) threshold has been exceeded by the estimated CPU load.

This governor exposes the following tunables:

freq_step Frequency step in percent of the maximum frequency the governor is allowed to set (the `scaling_max_freq` policy limit), between 0 and 100 (5 by default).

This is how much the frequency is allowed to change in one go. Setting it to 0 will cause the default frequency step (5 percent) to be used and setting it to 100 effectively causes the governor to periodically switch the frequency between the `scaling_min_freq` and `scaling_max_freq` policy limits.

down_threshold Threshold value (in percent, 20 by default) used to determine the frequency change direction.

If the estimated CPU load is greater than this value, the frequency will go up (by `freq_step`). If the load is less than this value (and the `sampling_down_factor` mechanism is not in effect), the frequency will go down. Otherwise, the frequency will not be changed.

sampling_down_factor Frequency decrease deferral factor, between 1 (default) and 10 inclusive.

It effectively causes the frequency to go down `sampling_down_factor` times slower than it ramps up.

Frequency Boost Support

Background

Some processors support a mechanism to raise the operating frequency of some cores in a multicore package temporarily (and above the sustainable frequency threshold for the whole package) under certain conditions, for example if the whole chip is not fully utilized and below its intended thermal or power budget.

Different names are used by different vendors to refer to this functionality. For Intel processors it is referred to as "Turbo Boost", AMD calls it "Turbo-Core" or (in technical documentation) "Core Performance Boost" and so on. As a rule, it also is implemented differently by different vendors. The simple term "frequency boost" is used here for brevity to refer to all of those implementations.

The frequency boost mechanism may be either hardware-based or software-based. If it is hardware-based (e.g. on x86), the decision to trigger the boosting is made by the hardware (although in general it requires the hardware to be put into a special state in which it can control the CPU frequency within certain limits). If it is software-based (e.g. on ARM), the scaling driver decides whether or not to trigger boosting and when to do that.

The `boost` File in `sysfs`

This file is located under `/sys/devices/system/cpu/cpufreq/` and controls the "boost" setting for the whole system. It is not present if the underlying scaling driver does not support the frequency boost mechanism (or supports it, but provides a driver-specific interface for controlling it, like *intel_pstate*).

If the value in this file is 1, the frequency boost mechanism is enabled. This means that either the hardware can be put into states in which it is able to trigger boosting (in the hardware-based case), or the software is allowed to trigger boosting (in the software-based case). It does not mean that boosting is actually in use at the moment on any CPUs in the system. It only means a permission to use the frequency boost mechanism (which still may never be used for other reasons).

If the value in this file is 0, the frequency boost mechanism is disabled and cannot be used at all.

The only values that can be written to this file are 0 and 1.

Rationale for Boost Control Knob

The frequency boost mechanism is generally intended to help to achieve optimum CPU performance on time scales below software resolution (e.g. below the scheduler tick interval) and it is demonstrably suitable for many workloads, but it may lead to problems in certain situations.

For this reason, many systems make it possible to disable the frequency boost mechanism in the platform firmware (BIOS) setup, but that requires the system to be restarted for the setting to be adjusted as desired, which may not be practical at least in some cases. For example:

1. Boosting means overclocking the processor, although under controlled conditions. Generally, the processor's energy consumption increases as a result of increasing its frequency and voltage, even temporarily. That may not be desirable on systems that switch to power sources of limited capacity, such as batteries, so the ability to disable the boost mechanism while the system is running may help there (but that depends on the workload too).
2. In some situations deterministic behavior is more important than performance or energy consumption (or both) and the ability to disable boosting while the system is running may be useful then.
3. To examine the impact of the frequency boost mechanism itself, it is useful to be able to run tests with and without boosting, preferably without restarting the system in the meantime.
4. Reproducible results are important when running benchmarks. Since the boosting functionality depends on the load of the whole package, single-thread performance may vary because of it which

may lead to unreproducible results sometimes. That can be avoided by disabling the frequency boost mechanism before running benchmarks sensitive to that issue.

Legacy AMD cpb Knob

The AMD powernow-k8 scaling driver supports a sysfs knob very similar to the global boost one. It is used for disabling/enabling the "Core Performance Boost" feature of some AMD processors.

If present, that knob is located in every CPUFreq policy directory in sysfs (/sys/devices/system/cpu/cpufreq/policyX/) and is called cpb, which indicates a more fine grained control interface. The actual implementation, however, works on the system-wide basis and setting that knob for one policy causes the same value of it to be set for all of the other policies at the same time.

That knob is still supported on AMD processors that support its underlying hardware feature, but it may be configured out of the kernel (via the CONFIG_X86_ACPI_CPUFREQ_CPB configuration option) and the global boost knob is present regardless. Thus it is always possible use the boost knob instead of the cpb one which is highly recommended, as that is more consistent with what all of the other systems do (and the cpb knob may not be supported any more in the future).

The cpb knob is never present for any processors without the underlying hardware feature (e.g. all Intel ones), even if the CONFIG_X86_ACPI_CPUFREQ_CPB configuration option is set.

intel_pstate CPU Performance Scaling Driver

```
Copyright (c) 2017 Intel Corp., Rafael J. Wysocki <rafael.j.wysocki@intel.com>
```

General Information

intel_pstate is a part of the *CPU performance scaling subsystem* in the Linux kernel (CPUFreq). It is a scaling driver for the Sandy Bridge and later generations of Intel processors. Note, however, that some of those processors may not be supported. [To understand intel_pstate it is necessary to know how CPUFreq works in general, so this is the time to read *CPU Performance Scaling* if you have not done that yet.]

For the processors supported by intel_pstate, the P-state concept is broader than just an operating frequency or an operating performance point (see the LinuxCon Europe 2015 presentation by Kristen Accardi for more information about that). For this reason, the representation of P-states used by intel_pstate internally follows the hardware specification (for details refer to Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3: System Programming Guide). However, the CPUFreq core uses frequencies for identifying operating performance points of CPUs and frequencies are involved in the user space interface exposed by it, so intel_pstate maps its internal representation of P-states to frequencies too (fortunately, that mapping is unambiguous). At the same time, it would not be practical for intel_pstate to supply the CPUFreq core with a table of available frequencies due to the possible size of it, so the driver does not do that. Some functionality of the core is limited by that.

Since the hardware P-state selection interface used by intel_pstate is available at the logical CPU level, the driver always works with individual CPUs. Consequently, if intel_pstate is in use, every CPUFreq policy object corresponds to one logical CPU and CPUFreq policies are effectively equivalent to CPUs. In particular, this means that they become "inactive" every time the corresponding CPU is taken offline and need to be re-initialized when it goes back online.

intel_pstate is not modular, so it cannot be unloaded, which means that the only way to pass early-configuration-time parameters to it is via the kernel command line. However, its configuration can be adjusted via sysfs to a great extent. In some configurations it even is possible to unregister it via sysfs which allows another CPUFreq scaling driver to be loaded and registered (see *below*).

Operation Modes

intel_pstate can operate in three different modes: in the active mode with or without hardware-managed P-states support and in the passive mode. Which of them will be in effect depends on what kernel command line options are used and on the capabilities of the processor.

Active Mode

This is the default operation mode of intel_pstate. If it works in this mode, the scaling_driver policy attribute in sysfs for all CPUFreq policies contains the string "intel_pstate".

In this mode the driver bypasses the scaling governors layer of CPUFreq and provides its own scaling algorithms for P-state selection. Those algorithms can be applied to CPUFreq policies in the same way as generic scaling governors (that is, through the scaling_governor policy attribute in sysfs). [Note that different P-state selection algorithms may be chosen for different policies, but that is not recommended.]

They are not generic scaling governors, but their names are the same as the names of some of those governors. Moreover, confusingly enough, they generally do not work in the same way as the generic governors they share the names with. For example, the powersave P-state selection algorithm provided by intel_pstate is not a counterpart of the generic powersave governor (roughly, it corresponds to the schedutil and ondemand governors).

There are two P-state selection algorithms provided by intel_pstate in the active mode: powersave and performance. The way they both operate depends on whether or not the hardware-managed P-states (HWP) feature has been enabled in the processor and possibly on the processor model.

Which of the P-state selection algorithms is used by default depends on the CONFIG_CPU_FREQ_DEFAULT_GOV_PERFORMANCE kernel configuration option. Namely, if that option is set, the performance algorithm will be used by default, and the other one will be used by default if it is not set.

Active Mode With HWP

If the processor supports the HWP feature, it will be enabled during the processor initialization and cannot be disabled after that. It is possible to avoid enabling it by passing the intel_pstate=no_hwp argument to the kernel in the command line.

If the HWP feature has been enabled, intel_pstate relies on the processor to select P-states by itself, but still it can give hints to the processor's internal P-state selection logic. What those hints are depends on which P-state selection algorithm has been applied to the given policy (or to the CPU it corresponds to).

Even though the P-state selection is carried out by the processor automatically, intel_pstate registers utilization update callbacks with the CPU scheduler in this mode. However, they are not used for running a P-state selection algorithm, but for periodic updates of the current CPU frequency information to be made available from the scaling_cur_freq policy attribute in sysfs.

HWP + performance

In this configuration intel_pstate will write 0 to the processor's Energy-Performance Preference (EPP) knob (if supported) or its Energy-Performance Bias (EPB) knob (otherwise), which means that the processor's internal P-state selection logic is expected to focus entirely on performance.

This will override the EPP/EPB setting coming from the sysfs interface (see *Energy vs Performance Hints* below).

Also, in this configuration the range of P-states available to the processor's internal P-state selection logic is always restricted to the upper boundary (that is, the maximum P-state that the driver is allowed to use).

HWP + powersave

In this configuration intel_pstate will set the processor's Energy-Performance Preference (EPP) knob (if supported) or its Energy-Performance Bias (EPB) knob (otherwise) to whatever value it was previously set to via sysfs (or whatever default value it was set to by the platform firmware). This usually causes the processor's internal P-state selection logic to be less performance-focused.

Active Mode Without HWP

This is the default operation mode for processors that do not support the HWP feature. It also is used by default with the intel_pstate=no_hwp argument in the kernel command line. However, in this mode intel_pstate may refuse to work with the given processor if it does not recognize it. [Note that intel_pstate will never refuse to work with any processor with the HWP feature enabled.]

In this mode intel_pstate registers utilization update callbacks with the CPU scheduler in order to run a P-state selection algorithm, either powersave or performance, depending on the scaling_cur_freq policy setting in sysfs. The current CPU frequency information to be made available from the scaling_cur_freq policy attribute in sysfs is periodically updated by those utilization update callbacks too.

performance

Without HWP, this P-state selection algorithm is always the same regardless of the processor model and platform configuration.

It selects the maximum P-state it is allowed to use, subject to limits set via sysfs, every time the driver configuration for the given CPU is updated (e.g. via sysfs).

This is the default P-state selection algorithm if the CONFIG_CPU_FREQ_DEFAULT_GOV_PERFORMANCE kernel configuration option is set.

powersave

Without HWP, this P-state selection algorithm is similar to the algorithm implemented by the generic schedutil scaling governor except that the utilization metric used by it is based on numbers coming from feedback registers of the CPU. It generally selects P-states proportional to the current CPU utilization.

This algorithm is run by the driver's utilization update callback for the given CPU when it is invoked by the CPU scheduler, but not more often than every 10 ms. Like in the performance case, the hardware configuration is not touched if the new P-state turns out to be the same as the current one.

This is the default P-state selection algorithm if the CONFIG_CPU_FREQ_DEFAULT_GOV_PERFORMANCE kernel configuration option is not set.

Passive Mode

This mode is used if the intel_pstate=passive argument is passed to the kernel in the command line (it implies the intel_pstate=no_hwp setting too). Like in the active mode without HWP support, in this mode intel_pstate may refuse to work with the given processor if it does not recognize it.

If the driver works in this mode, the scaling_driver policy attribute in sysfs for all CPUFreq policies contains the string "intel_cpufreq". Then, the driver behaves like a regular CPUFreq scaling driver. That is, it is invoked by generic scaling governors when necessary to talk to the hardware in order to change the P-state of a CPU (in particular, the schedutil governor can invoke it directly from scheduler context).

While in this mode, intel_pstate can be used with all of the (generic) scaling governors listed by the scaling_available_governors policy attribute in sysfs (and the P-state selection algorithms described above are not used). Then, it is responsible for the configuration of policy objects corresponding to CPUs

and provides the CPUFreq core (and the scaling governors attached to the policy objects) with accurate information on the maximum and minimum operating frequencies supported by the hardware (including the so-called "turbo" frequency ranges). In other words, in the passive mode the entire range of available P-states is exposed by `intel_pstate` to the CPUFreq core. However, in this mode the driver does not register utilization update callbacks with the CPU scheduler and the `scaling_cur_freq` information comes from the CPUFreq core (and is the last frequency selected by the current scaling governor for the given policy).

Turbo P-states Support

In the majority of cases, the entire range of P-states available to `intel_pstate` can be divided into two sub-ranges that correspond to different types of processor behavior, above and below a boundary that will be referred to as the "turbo threshold" in what follows.

The P-states above the turbo threshold are referred to as "turbo P-states" and the whole sub-range of P-states they belong to is referred to as the "turbo range". These names are related to the Turbo Boost technology allowing a multicore processor to opportunistically increase the P-state of one or more cores if there is enough power to do that and if that is not going to cause the thermal envelope of the processor package to be exceeded.

Specifically, if software sets the P-state of a CPU core within the turbo range (that is, above the turbo threshold), the processor is permitted to take over performance scaling control for that core and put it into turbo P-states of its choice going forward. However, that permission is interpreted differently by different processor generations. Namely, the Sandy Bridge generation of processors will never use any P-states above the last one set by software for the given core, even if it is within the turbo range, whereas all of the later processor generations will take it as a license to use any P-states from the turbo range, even above the one set by software. In other words, on those processors setting any P-state from the turbo range will enable the processor to put the given core into all turbo P-states up to and including the maximum supported one as it sees fit.

One important property of turbo P-states is that they are not sustainable. More precisely, there is no guarantee that any CPUs will be able to stay in any of those states indefinitely, because the power distribution within the processor package may change over time or the thermal envelope it was designed for might be exceeded if a turbo P-state was used for too long.

In turn, the P-states below the turbo threshold generally are sustainable. In fact, if one of them is set by software, the processor is not expected to change it to a lower one unless in a thermal stress or a power limit violation situation (a higher P-state may still be used if it is set for another CPU in the same package at the same time, for example).

Some processors allow multiple cores to be in turbo P-states at the same time, but the maximum P-state that can be set for them generally depends on the number of cores running concurrently. The maximum turbo P-state that can be set for 3 cores at the same time usually is lower than the analogous maximum P-state for 2 cores, which in turn usually is lower than the maximum turbo P-state that can be set for 1 core. The one-core maximum turbo P-state is thus the maximum supported one overall.

The maximum supported turbo P-state, the turbo threshold (the maximum supported non-turbo P-state) and the minimum supported P-state are specific to the processor model and can be determined by reading the processor's model-specific registers (MSRs). Moreover, some processors support the Configurable TDP (Thermal Design Power) feature and, when that feature is enabled, the turbo threshold effectively becomes a configurable value that can be set by the platform firmware.

Unlike _PSS objects in the ACPI tables, `intel_pstate` always exposes the entire range of available P-states, including the whole turbo range, to the CPUFreq core and (in the passive mode) to generic scaling governors. This generally causes turbo P-states to be set more often when `intel_pstate` is used relative to ACPI-based CPU performance scaling (see *below* for more information).

Moreover, since `intel_pstate` always knows what the real turbo threshold is (even if the Configurable TDP feature is enabled in the processor), its `no_turbo` attribute in `sysfs` (described *below*) should work as expected in all cases (that is, if set to disable turbo P-states, it always should prevent `intel_pstate` from using them).

Processor Support

To handle a given processor `intel_pstate` requires a number of different pieces of information on it to be known, including:

- The minimum supported P-state.
- The maximum supported *non-turbo P-state*.
- Whether or not turbo P-states are supported at all.
- The maximum supported *one-core turbo P-state* (if turbo P-states are supported).
- The scaling formula to translate the driver's internal representation of P-states into frequencies and the other way around.

Generally, ways to obtain that information are specific to the processor model or family. Although it often is possible to obtain all of it from the processor itself (using model-specific registers), there are cases in which hardware manuals need to be consulted to get to it too.

For this reason, there is a list of supported processors in `intel_pstate` and the driver initialization will fail if the detected processor is not in that list, unless it supports the *HWP feature*. [The interface to obtain all of the information listed above is the same for all of the processors supporting the HWP feature, which is why they all are supported by `intel_pstate`.]

User Space Interface in `sysfs`

Global Attributes

`intel_pstate` exposes several global attributes (files) in `sysfs` to control its functionality at the system level. They are located in the `/sys/devices/system/cpu/cpufreq/intel_pstate/` directory and affect all CPUs.

Some of them are not present if the `intel_pstate=per_cpu_perf_limits` argument is passed to the kernel in the command line.

`max_perf_pct` Maximum P-state the driver is allowed to set in percent of the maximum supported performance level (the highest supported *turbo P-state*).

This attribute will not be exposed if the `intel_pstate=per_cpu_perf_limits` argument is present in the kernel command line.

`min_perf_pct` Minimum P-state the driver is allowed to set in percent of the maximum supported performance level (the highest supported *turbo P-state*).

This attribute will not be exposed if the `intel_pstate=per_cpu_perf_limits` argument is present in the kernel command line.

`num_pstates` Number of P-states supported by the processor (between 0 and 255 inclusive) including both turbo and non-turbo P-states (see *Turbo P-states Support*).

The value of this attribute is not affected by the `no_turbo` setting described *below*.

This attribute is read-only.

`turbo_pct` Ratio of the *turbo range* size to the size of the entire range of supported P-states, in percent.

This attribute is read-only.

`no_turbo` If set (equal to 1), the driver is not allowed to set any turbo P-states (see *Turbo P-states Support*). If unset (equalt to 0, which is the default), turbo P-states can be set by the driver. [Note that `intel_pstate` does not support the general `boost` attribute (supported by some other scaling drivers) which is replaced by this one.]

This attrubute does not affect the maximum supported frequency value supplied to the `CPUFreq` core and exposed via the policy interface, but it affects the maximum possible value of per-policy P-state limits (see *Interpretation of Policy Attributes* below for details).

status Operation mode of the driver: “active”, “passive” or “off”.

“active” The driver is functional and in the *active mode*.

“passive” The driver is functional and in the *passive mode*.

“off” The driver is not functional (it is not registered as a scaling driver with the CPUFreq core).

This attribute can be written to in order to change the driver’s operation mode or to unregister it. The string written to it must be one of the possible values of it and, if successful, the write will cause the driver to switch over to the operation mode represented by that string - or to be unregistered in the “off” case. [Actually, switching over from the active mode to the passive mode or the other way around causes the driver to be unregistered and registered again with a different set of callbacks, so all of its settings (the global as well as the per-policy ones) are then reset to their default values, possibly depending on the target operation mode.]

That only is supported in some configurations, though (for example, if the *HWP feature is enabled in the processor*, the operation mode of the driver cannot be changed), and if it is not supported in the current configuration, writes to this attribute with fail with an appropriate error.

Interpretation of Policy Attributes

The interpretation of some CPUFreq policy attributes described in *CPU Performance Scaling* is special with `intel_pstate` as the current scaling driver and it generally depends on the driver’s *operation mode*.

First of all, the values of the `cpuinfo_max_freq`, `cpuinfo_min_freq` and `scaling_cur_freq` attributes are produced by applying a processor-specific multiplier to the internal P-state representation used by `intel_pstate`. Also, the values of the `scaling_max_freq` and `scaling_min_freq` attributes are capped by the frequency corresponding to the maximum P-state that the driver is allowed to set.

If the `no_turbo` *global attribute* is set, the driver is not allowed to use turbo P-states, so the maximum value of `scaling_max_freq` and `scaling_min_freq` is limited to the maximum non-turbo P-state frequency. Accordingly, setting `no_turbo` causes `scaling_max_freq` and `scaling_min_freq` to go down to that value if they were above it before. However, the old values of `scaling_max_freq` and `scaling_min_freq` will be restored after unsetting `no_turbo`, unless these attributes have been written to after `no_turbo` was set.

If `no_turbo` is not set, the maximum possible value of `scaling_max_freq` and `scaling_min_freq` corresponds to the maximum supported turbo P-state, which also is the value of `cpuinfo_max_freq` in either case.

Next, the following policy attributes have special meaning if `intel_pstate` works in the *active mode*:

`scaling_available_governors` List of P-state selection algorithms provided by `intel_pstate`.

`scaling_governor` P-state selection algorithm provided by `intel_pstate` currently in use with the given policy.

`scaling_cur_freq` Frequency of the average P-state of the CPU represented by the given policy for the time interval between the last two invocations of the driver’s utilization update callback by the CPU scheduler for that CPU.

The meaning of these attributes in the *passive mode* is the same as for other scaling drivers.

Additionally, the value of the `scaling_driver` attribute for `intel_pstate` depends on the operation mode of the driver. Namely, it is either “intel_pstate” (in the *active mode*) or “intel_cpufreq” (in the *passive mode*).

Coordination of P-State Limits

`intel_pstate` allows P-state limits to be set in two ways: with the help of the `max_perf_pct` and `min_perf_pct` *global attributes* or via the `scaling_max_freq` and `scaling_min_freq` CPUFreq policy at-

tributes. The coordination between those limits is based on the following rules, regardless of the current operation mode of the driver:

1. All CPUs are affected by the global limits (that is, none of them can be requested to run faster than the global maximum and none of them can be requested to run slower than the global minimum).
2. Each individual CPU is affected by its own per-policy limits (that is, it cannot be requested to run faster than its own per-policy maximum and it cannot be requested to run slower than its own per-policy minimum).
3. The global and per-policy limits can be set independently.

If the *HWP feature is enabled in the processor*, the resulting effective values are written into its registers whenever the limits change in order to request its internal P-state selection logic to always set P-states within these limits. Otherwise, the limits are taken into account by scaling governors (in the *passive mode*) and by the driver every time before setting a new P-state for a CPU.

Additionally, if the `intel_pstate=per_cpu_perf_limits` command line argument is passed to the kernel, `max_perf_pct` and `min_perf_pct` are not exposed at all and the only way to set the limits is by using the policy attributes.

Energy vs Performance Hints

If `intel_pstate` works in the *active mode with the HWP feature enabled* in the processor, additional attributes are present in every `CPUFreq` policy directory in `sysfs`. They are intended to allow user space to help `intel_pstate` to adjust the processor's internal P-state selection logic by focusing it on performance or on energy-efficiency, or somewhere between the two extremes:

energy_performance_preference Current value of the energy vs performance hint for the given policy (or the CPU represented by it).

The hint can be changed by writing to this attribute.

energy_performance_available_preferences List of strings that can be written to the `energy_performance_preference` attribute.

They represent different energy vs performance hints and should be self-explanatory, except that `default` represents whatever hint value was set by the platform firmware.

Strings written to the `energy_performance_preference` attribute are internally translated to integer values written to the processor's Energy-Performance Preference (EPP) knob (if supported) or its Energy-Performance Bias (EPB) knob.

[Note that tasks may by migrated from one CPU to another by the scheduler's load-balancing algorithm and if different energy vs performance hints are set for those CPUs, that may lead to undesirable outcomes. To avoid such issues it is better to set the same energy vs performance hint for all CPUs or to pin every task potentially sensitive to them to a specific CPU.]

`intel_pstate` vs `acpi-cpufreq`

On the majority of systems supported by `intel_pstate`, the ACPI tables provided by the platform firmware contain _PSS objects returning information that can be used for CPU performance scaling (refer to the ACPI specification for details on the _PSS objects and the format of the information returned by them).

The information returned by the ACPI _PSS objects is used by the `acpi-cpufreq` scaling driver. On systems supported by `intel_pstate` the `acpi-cpufreq` driver uses the same hardware CPU performance scaling interface, but the set of P-states it can use is limited by the _PSS output.

On those systems each _PSS object returns a list of P-states supported by the corresponding CPU which basically is a subset of the P-states range that can be used by `intel_pstate` on the same system, with one exception: the whole *turbo range* is represented by one item in it (the topmost one). By convention, the frequency returned by _PSS for that item is greater by 1 MHz than the frequency of the highest

non-turbo P-state listed by it, but the corresponding P-state representation (following the hardware specification) returned for it matches the maximum supported turbo P-state (or is the special value 255 meaning essentially "go as high as you can get").

The list of P-states returned by _PSS is reflected by the table of available frequencies supplied by `acpi-cpufreq` to the `CPUFreq` core and scaling governors and the minimum and maximum supported frequencies reported by it come from that list as well. In particular, given the special representation of the turbo range described above, this means that the maximum supported frequency reported by `acpi-cpufreq` is higher by 1 MHz than the frequency of the highest supported non-turbo P-state listed by _PSS which, of course, affects decisions made by the scaling governors, except for `powersave` and `performance`.

For example, if a given governor attempts to select a frequency proportional to estimated CPU load and maps the load of 100% to the maximum supported frequency (possibly multiplied by a constant), then it will tend to choose P-states below the turbo threshold if `acpi-cpufreq` is used as the scaling driver, because in that case the turbo range corresponds to a small fraction of the frequency band it can use (1 MHz vs 1 GHz or more). In consequence, it will only go to the turbo range for the highest loads and the other loads above 50% that might benefit from running at turbo frequencies will be given non-turbo P-states instead.

One more issue related to that may appear on systems supporting the *Configurable TDP feature* allowing the platform firmware to set the turbo threshold. Namely, if that is not coordinated with the lists of P-states returned by _PSS properly, there may be more than one item corresponding to a turbo P-state in those lists and there may be a problem with avoiding the turbo range (if desirable or necessary). Usually, to avoid using turbo P-states overall, `acpi-cpufreq` simply avoids using the topmost state listed by _PSS, but that is not sufficient when there are other turbo P-states in the list returned by it.

Apart from the above, `acpi-cpufreq` works like `intel_pstate` in the *passive mode*, except that the number of P-states it can set is limited to the ones listed by the ACPI _PSS objects.

Kernel Command Line Options for `intel_pstate`

Several kernel command line options can be used to pass early-configuration-time parameters to `intel_pstate` in order to enforce specific behavior of it. All of them have to be prepended with the `intel_pstate=` prefix.

`disable` Do not register `intel_pstate` as the scaling driver even if the processor is supported by it.

`passive` Register `intel_pstate` in the *passive mode* to start with.

This option implies the `no_hwp` one described below.

`force` Register `intel_pstate` as the scaling driver instead of `acpi-cpufreq` even if the latter is preferred on the given system.

This may prevent some platform features (such as thermal controls and power capping) that rely on the availability of ACPI P-states information from functioning as expected, so it should be used with caution.

This option does not work with processors that are not supported by `intel_pstate` and on platforms where the `pcc-cpufreq` scaling driver is used instead of `acpi-cpufreq`.

`no_hwp` Do not enable the *hardware-managed P-states (HWP) feature* even if it is supported by the processor.

`hwp_only` Register `intel_pstate` as the scaling driver only if the *hardware-managed P-states (HWP) feature* is supported by the processor.

`support_acpi_ppc` Take ACPI _PPC performance limits into account.

If the preferred power management profile in the FADT (Fixed ACPI Description Table) is set to "Enterprise Server" or "Performance Server", the ACPI _PPC limits are taken into account by default and this option has no effect.

`per_cpu_perf_limits` Use per-logical-CPU P-State limits (see *Coordination of P-state Limits* for details).

Diagnostics and Tuning

Trace Events

There are two static trace events that can be used for `intel_pstate` diagnostics. One of them is the `cpu_frequency` trace event generally used by CPUFreq, and the other one is the `pstate_sample` trace event specific to `intel_pstate`. Both of them are triggered by `intel_pstate` only if it works in the *active mode*.

The following sequence of shell commands can be used to enable them and see their output (if the kernel is generally configured to support event tracing):

```
# cd /sys/kernel/debug/tracing/
# echo 1 > events/power/pstate_sample/enable
# echo 1 > events/power/cpu_frequency/enable
# cat trace
gnome-terminal--4510  [001] ..s.  1177.680733: pstate_sample: core_busy=107 scaled=94 from=26␣
 ↪to=26 mperf=1143818 aperf=1230607 tsc=29838618 freq=2474476
cat-5235  [002] ..s.  1177.681723: cpu_frequency: state=2900000 cpu_id=2
```

If `intel_pstate` works in the *passive mode*, the `cpu_frequency` trace event will be triggered either by the `schedutil` scaling governor (for the policies it is attached to), or by the CPUFreq core (for the policies with other scaling governors).

`ftrace`

The `ftrace` interface can be used for low-level diagnostics of `intel_pstate`. For example, to check how often the function to set a P-state is called, the `ftrace` filter can be set to to `intel_pstate_set_pstate()`:

```
# cd /sys/kernel/debug/tracing/
# cat available_filter_functions | grep -i pstate
intel_pstate_set_pstate
intel_pstate_cpu_init
...
# echo intel_pstate_set_pstate > set_ftrace_filter
# echo function > current_tracer
# cat trace | head -15
# tracer: function
#
# entries-in-buffer/entries-written: 80/80   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
            Xorg-3129  [000] ..s.  2537.644844: intel_pstate_set_pstate <-intel_pstate_timer_
 ↪func
 gnome-terminal--4510  [002] ..s.  2537.649844: intel_pstate_set_pstate <-intel_pstate_timer_
 ↪func
     gnome-shell-3409  [001] ..s.  2537.650850: intel_pstate_set_pstate <-intel_pstate_timer_
 ↪func
          <idle>-0     [000] ..s.  2537.654843: intel_pstate_set_pstate <-intel_pstate_timer_
 ↪func
```

CHAPTER

TWENTYSEVEN

THUNDERBOLT

The interface presented here is not meant for end users. Instead there should be a userspace tool that handles all the low-level details, keeps database of the authorized devices and prompts user for new connections.

More details about the sysfs interface for Thunderbolt devices can be found in `Documentation/ABI/testing/sysfs-bus-thunderbolt`.

Those users who just want to connect any device without any sort of manual work, can add following line to `/etc/udev/rules.d/99-local.rules`:

```
ACTION=="add", SUBSYSTEM=="thunderbolt", ATTR{authorized}=="0", ATTR{authorized}="1"
```

This will authorize all devices automatically when they appear. However, keep in mind that this bypasses the security levels and makes the system vulnerable to DMA attacks.

Security levels and how to use them

Starting from Intel Falcon Ridge Thunderbolt controller there are 4 security levels available. The reason for these is the fact that the connected devices can be DMA masters and thus read contents of the host memory without CPU and OS knowing about it. There are ways to prevent this by setting up an IOMMU but it is not always available for various reasons.

The security levels are as follows:

none All devices are automatically connected by the firmware. No user approval is needed. In BIOS settings this is typically called *Legacy mode*.

user User is asked whether the device is allowed to be connected. Based on the device identification information available through `/sys/bus/thunderbolt/devices`. user then can do the decision. In BIOS settings this is typically called *Unique ID*.

secure User is asked whether the device is allowed to be connected. In addition to UUID the device (if it supports secure connect) is sent a challenge that should match the expected one based on a random key written to key sysfs attribute. In BIOS settings this is typically called *One time saved key*.

dponly The firmware automatically creates tunnels for Display Port and USB. No PCIe tunneling is done. In BIOS settings this is typically called *Display Port Only*.

The current security level can be read from `/sys/bus/thunderbolt/devices/domainX/security` where `domainX` is the Thunderbolt domain the host controller manages. There is typically one domain per Thunderbolt host controller.

If the security level reads as `user` or `secure` the connected device must be authorized by the user before PCIe tunnels are created (e.g the PCIe device appears).

Each Thunderbolt device plugged in will appear in sysfs under `/sys/bus/thunderbolt/devices`. The device directory carries information that can be used to identify the particular device, including its name and UUID.

Authorizing devices when security level is user or secure

When a device is plugged in it will appear in sysfs as follows:

```
/sys/bus/thunderbolt/devices/0-1/authorized   - 0
/sys/bus/thunderbolt/devices/0-1/device       - 0x8004
/sys/bus/thunderbolt/devices/0-1/device_name  - Thunderbolt to FireWire Adapter
/sys/bus/thunderbolt/devices/0-1/vendor       - 0x1
/sys/bus/thunderbolt/devices/0-1/vendor_name  - Apple, Inc.
/sys/bus/thunderbolt/devices/0-1/unique_id    - e0376f00-0300-0100-ffff-ffffffffffff
```

The authorized attribute reads 0 which means no PCIe tunnels are created yet. The user can authorize the device by simply:

```
# echo 1 > /sys/bus/thunderbolt/devices/0-1/authorized
```

This will create the PCIe tunnels and the device is now connected.

If the device supports secure connect, and the domain security level is set to secure, it has an additional attribute key which can hold a random 32 byte value used for authorization and challenging the device in future connects:

```
/sys/bus/thunderbolt/devices/0-3/authorized   - 0
/sys/bus/thunderbolt/devices/0-3/device       - 0x305
/sys/bus/thunderbolt/devices/0-3/device_name  - AKiTiO Thunder3 PCIe Box
/sys/bus/thunderbolt/devices/0-3/key          -
/sys/bus/thunderbolt/devices/0-3/vendor       - 0x41
/sys/bus/thunderbolt/devices/0-3/vendor_name  - inXtron
/sys/bus/thunderbolt/devices/0-3/unique_id    - dc010000-0000-8508-a22d-32ca6421cb16
```

Notice the key is empty by default.

If the user does not want to use secure connect it can just echo 1 to the authorized attribute and the PCIe tunnels will be created in the same way than in user security level.

If the user wants to use secure connect, the first time the device is plugged a key needs to be created and send to the device:

```
# key=$(openssl rand -hex 32)
# echo $key > /sys/bus/thunderbolt/devices/0-3/key
# echo 1 > /sys/bus/thunderbolt/devices/0-3/authorized
```

Now the device is connected (PCIe tunnels are created) and in addition the key is stored on the device NVM.

Next time the device is plugged in the user can verify (challenge) the device using the same key:

```
# echo $key > /sys/bus/thunderbolt/devices/0-3/key
# echo 2 > /sys/bus/thunderbolt/devices/0-3/authorized
```

If the challenge the device returns back matches the one we expect based on the key, the device is connected and the PCIe tunnels are created. However, if the challenge failed no tunnels are created and error is returned to the user.

If the user still wants to connect the device it can either approve the device without a key or write new key and write 1 to the authorized file to get the new key stored on the device NVM.

Upgrading NVM on Thunderbolt device or host

Since most of the functionality is handled in a firmware running on a host controller or a device, it is important that the firmware can be upgraded to the latest where possible bugs in it have been fixed.

Typically OEMs provide this firmware from their support site.

There is also a central site which has links where to download firmwares for some machines:

Thunderbolt Updates

Before you upgrade firmware on a device or host, please make sure it is the suitable. Failing to do that may render the device (or host) in a state where it cannot be used properly anymore without special tools!

Host NVM upgrade on Apple Macs is not supported.

Once the NVM image has been downloaded, you need to plug in a Thunderbolt device so that the host controller appears. It does not matter which device is connected (unless you are upgrading NVM on a device - then you need to connect that particular device).

Note OEM-specific method to power the controller up ("force power") may be available for your system in which case there is no need to plug in a Thunderbolt device.

After that we can write the firmware to the non-active parts of the NVM of the host or device. As an example here is how Intel NUC6i7KYK (Skull Canyon) Thunderbolt controller NVM is upgraded:

```
# dd if=KYK_TBT_FW_0018.bin of=/sys/bus/thunderbolt/devices/0-0/nvm_non_active0/nvmem
```

Once the operation completes we can trigger NVM authentication and upgrade process as follows:

```
# echo 1 > /sys/bus/thunderbolt/devices/0-0/nvm_authenticate
```

If no errors are returned, the host controller shortly disappears. Once it comes back the driver notices it and initiates a full power cycle. After a while the host controller appears again and this time it should be fully functional.

We can verify that the new NVM firmware is active by running following commands:

```
# cat /sys/bus/thunderbolt/devices/0-0/nvm_authenticate
0x0
# cat /sys/bus/thunderbolt/devices/0-0/nvm_version
18.0
```

If `nvm_authenticate` contains anything else than 0x0 it is the error code from the last authentication cycle, which means the authentication of the NVM image failed.

Note names of the NVMem devices `nvm_activeN` and `nvm_non_activeN` depends on the order they are registered in the NVMem subsystem. N in the name is the identifier added by the NVMem subsystem.

Upgrading NVM when host controller is in safe mode

If the existing NVM is not properly authenticated (or is missing) the host controller goes into safe mode which means that only available functionality is flashing new NVM image. When in this mode the reading `nvm_version` fails with `ENODATA` and the device identification information is missing.

To recover from this mode, one needs to flash a valid NVM image to the host host controller in the same way it is done in the previous chapter.

CHAPTER

TWENTYEIGHT

LINUX SECURITY MODULE USAGE

The Linux Security Module (LSM) framework provides a mechanism for various security checks to be hooked by new kernel extensions. The name “module” is a bit of a misnomer since these extensions are not actually loadable kernel modules. Instead, they are selectable at build-time via CONFIG_DEFAULT_SECURITY and can be overridden at boot-time via the `"security=..."` kernel command line argument, in the case where multiple LSMs were built into a given kernel.

The primary users of the LSM interface are Mandatory Access Control (MAC) extensions which provide a comprehensive security policy. Examples include SELinux, Smack, Tomoyo, and AppArmor. In addition to the larger MAC extensions, other extensions can be built using the LSM to provide specific changes to system operation when these tweaks are not available in the core functionality of Linux itself.

Without a specific LSM built into the kernel, the default LSM will be the Linux capabilities system. Most LSMs choose to extend the capabilities system, building their checks on top of the defined capability hooks. For more details on capabilities, see `capabilities(7)` in the Linux man-pages project.

A list of the active security modules can be found by reading `/sys/kernel/security/lsm`. This is a comma separated list, and will always include the capability module. The list reflects the order in which checks are made. The capability module will always be first, followed by any “minor” modules (e.g. Yama) and then the one “major” module (e.g. SELinux) if there is one configured.

AppArmor

What is AppArmor?

AppArmor is MAC style security extension for the Linux kernel. It implements a task centered policy, with task “profiles” being created and loaded from user space. Tasks on the system that do not have a profile defined for them run in an unconfined state which is equivalent to standard Linux DAC permissions.

How to enable/disable

set `CONFIG_SECURITY_APPARMOR=y`

If AppArmor should be selected as the default security module then set:

```
CONFIG_DEFAULT_SECURITY="apparmor"
CONFIG_SECURITY_APPARMOR_BOOTPARAM_VALUE=1
```

Build the kernel

If AppArmor is not the default security module it can be enabled by passing `security=apparmor` on the kernel’s command line.

If AppArmor is the default security module it can be disabled by passing `apparmor=0,security=XXXX` (where XXXX is valid security module), on the kernel’s command line.

For AppArmor to enforce any restrictions beyond standard Linux DAC permissions policy must be loaded into the kernel from user space (see the Documentation and tools links).

Documentation

Documentation can be found on the wiki, linked below.

Links

Mailing List - apparmor@lists.ubuntu.com

Wiki - http://apparmor.wiki.kernel.org/

User space tools - https://launchpad.net/apparmor

Kernel module - git://git.kernel.org/pub/scm/linux/kernel/git/jj/apparmor-dev.git

LoadPin

LoadPin is a Linux Security Module that ensures all kernel-loaded files (modules, firmware, etc) all originate from the same filesystem, with the expectation that such a filesystem is backed by a read-only device such as dm-verity or CDROM. This allows systems that have a verified and/or unchangeable filesystem to enforce module and firmware loading restrictions without needing to sign the files individually.

The LSM is selectable at build-time with `CONFIG_SECURITY_LOADPIN`, and can be controlled at boot-time with the kernel command line option "`loadpin.enabled`". By default, it is enabled, but can be disabled at boot ("`loadpin.enabled=0`").

LoadPin starts pinning when it sees the first file loaded. If the block device backing the filesystem is not read-only, a sysctl is created to toggle pinning: `/proc/sys/kernel/loadpin/enabled`. (Having a mutable filesystem means pinning is mutable too, but having the sysctl allows for easy testing on systems with a mutable filesystem.)

SELinux

If you want to use SELinux, chances are you will want to use the distro-provided policies, or install the latest reference policy release from

> http://oss.tresys.com/projects/refpolicy

However, if you want to install a dummy policy for testing, you can do using `mdp` provided under scripts/selinux. Note that this requires the selinux userspace to be installed - in particular you will need checkpolicy to compile a kernel, and setfiles and fixfiles to label the filesystem.

1. Compile the kernel with selinux enabled.
2. Type `make` to compile `mdp`.
3. Make sure that you are not running with SELinux enabled and a real policy. If you are, reboot with selinux disabled before continuing.
4. Run install_policy.sh:

```
cd scripts/selinux
sh install_policy.sh
```

Step 4 will create a new dummy policy valid for your kernel, with a single selinux user, role, and type. It will compile the policy, will set your `SELINUXTYPE` to `dummy` in `/etc/selinux/config`, install the compiled policy as `dummy`, and relabel your filesystem.

Smack

> "Good for you, you've decided to clean the elevator!" - The Elevator, from Dark Star

Smack is the Simplified Mandatory Access Control Kernel. Smack is a kernel based implementation of mandatory access control that includes simplicity in its primary design goals.

Smack is not the only Mandatory Access Control scheme available for Linux. Those new to Mandatory Access Control are encouraged to compare Smack with the other mechanisms available to determine which is best suited to the problem at hand.

Smack consists of three major components:

- The kernel
- Basic utilities, which are helpful but not required
- Configuration data

The kernel component of Smack is implemented as a Linux Security Modules (LSM) module. It requires netlabel and works best with file systems that support extended attributes, although xattr support is not strictly required. It is safe to run a Smack kernel under a "vanilla" distribution.

Smack kernels use the CIPSO IP option. Some network configurations are intolerant of IP options and can impede access to systems that use them as Smack does.

Smack is used in the Tizen operating system. Please go to http://wiki.tizen.org for information about how Smack is used in Tizen.

The current git repository for Smack user space is:

git://github.com/smack-team/smack.git

This should make and install on most modern distributions. There are five commands included in smackutil:

chsmack: display or set Smack extended attribute values

smackctl: load the Smack access rules

smackaccess: report if a process with one label has access to an object with another

These two commands are obsolete with the introduction of the smackfs/load2 and smackfs/cipso2 interfaces.

smackload: properly formats data for writing to smackfs/load

smackcipso: properly formats data for writing to smackfs/cipso

In keeping with the intent of Smack, configuration data is minimal and not strictly required. The most important configuration step is mounting the smackfs pseudo filesystem. If smackutil is installed the startup script will take care of this, but it can be manually as well.

Add this line to `/etc/fstab`:

```
smackfs /sys/fs/smackfs smackfs defaults 0 0
```

The `/sys/fs/smackfs` directory is created by the kernel.

Smack uses extended attributes (xattrs) to store labels on filesystem objects. The attributes are stored in the extended attribute security name space. A process must have `CAP_MAC_ADMIN` to change any of these attributes.

The extended attributes that Smack uses are:

SMACK64 Used to make access control decisions. In almost all cases the label given to a new filesystem object will be the label of the process that created it.

SMACK64EXEC The Smack label of a process that execs a program file with this attribute set will run with this attribute's value.

SMACK64MMAP Don't allow the file to be mmapped by a process whose Smack label does not allow all of the access permitted to a process with the label contained in this attribute. This is a very specific use case for shared libraries.

SMACK64TRANSMUTE Can only have the value "TRUE". If this attribute is present on a directory when an object is created in the directory and the Smack rule (more below) that permitted the write access to the directory includes the transmute ("t") mode the object gets the label of the directory instead of the label of the creating process. If the object being created is a directory the SMACK64TRANSMUTE attribute is set as well.

SMACK64IPIN This attribute is only available on file descriptors for sockets. Use the Smack label in this attribute for access control decisions on packets being delivered to this socket.

SMACK64IPOUT This attribute is only available on file descriptors for sockets. Use the Smack label in this attribute for access control decisions on packets coming from this socket.

There are multiple ways to set a Smack label on a file:

```
# attr -S -s SMACK64 -V "value" path
# chsmack -a value path
```

A process can see the Smack label it is running with by reading `/proc/self/attr/current`. A process with `CAP_MAC_ADMIN` can set the process Smack by writing there.

Most Smack configuration is accomplished by writing to files in the smackfs filesystem. This pseudo-filesystem is mounted on `/sys/fs/smackfs`.

access Provided for backward compatibility. The access2 interface is preferred and should be used instead. This interface reports whether a subject with the specified Smack label has a particular access to an object with a specified Smack label. Write a fixed format access rule to this file. The next read will indicate whether the access would be permitted. The text will be either "1" indicating access, or "0" indicating denial.

access2 This interface reports whether a subject with the specified Smack label has a particular access to an object with a specified Smack label. Write a long format access rule to this file. The next read will indicate whether the access would be permitted. The text will be either "1" indicating access, or "0" indicating denial.

ambient This contains the Smack label applied to unlabeled network packets.

change-rule This interface allows modification of existing access control rules. The format accepted on write is:

```
"%s %s %s %s"
```

where the first string is the subject label, the second the object label, the third the access to allow and the fourth the access to deny. The access strings may contain only the characters "rwxat-". If a rule for a given subject and object exists it will be modified by enabling the permissions in the third string and disabling those in the fourth string. If there is no such rule it will be created using the access specified in the third and the fourth strings.

cipso Provided for backward compatibility. The cipso2 interface is preferred and should be used instead. This interface allows a specific CIPSO header to be assigned to a Smack label. The format accepted on write is:

```
"%24s%4d%4d"["%4d"]...
```

The first string is a fixed Smack label. The first number is the level to use. The second number is the number of categories. The following numbers are the categories:

```
"level-3-cats-5-19          3   2   5  19"
```

cipso2 This interface allows a specific CIPSO header to be assigned to a Smack label. The format accepted on write is:

```
"%s%4d%4d"["%4d"]...
```

The first string is a long Smack label. The first number is the level to use. The second number is the number of categories. The following numbers are the categories:

```
"level-3-cats-5-19   3   2   5  19"
```

direct This contains the CIPSO level used for Smack direct label representation in network packets.

doi This contains the CIPSO domain of interpretation used in network packets.

ipv6host This interface allows specific IPv6 internet addresses to be treated as single label hosts. Packets are sent to single label hosts only from processes that have Smack write access to the host label. All packets received from single label hosts are given the specified label. The format accepted on write is:

```
"%h:%h:%h:%h:%h:%h:%h:%h label" or
"%h:%h:%h:%h:%h:%h:%h:%h/%d label".
```

The "::" address shortcut is not supported. If label is "-DELETE" a matched entry will be deleted.

load Provided for backward compatibility. The load2 interface is preferred and should be used instead. This interface allows access control rules in addition to the system defined rules to be specified. The format accepted on write is:

```
"%24s%24s%5s"
```

where the first string is the subject label, the second the object label, and the third the requested access. The access string may contain only the characters "rwxat-", and specifies which sort of access is allowed. The "-" is a placeholder for permissions that are not allowed. The string "r-x–" would specify read and execute access. Labels are limited to 23 characters in length.

load2 This interface allows access control rules in addition to the system defined rules to be specified. The format accepted on write is:

```
"%s %s %s"
```

where the first string is the subject label, the second the object label, and the third the requested access. The access string may contain only the characters "rwxat-", and specifies which sort of access is allowed. The "-" is a placeholder for permissions that are not allowed. The string "r-x–" would specify read and execute access.

load-self Provided for backward compatibility. The load-self2 interface is preferred and should be used instead. This interface allows process specific access rules to be defined. These rules are only consulted if access would otherwise be permitted, and are intended to provide additional restrictions on the process. The format is the same as for the load interface.

load-self2 This interface allows process specific access rules to be defined. These rules are only consulted if access would otherwise be permitted, and are intended to provide additional restrictions on the process. The format is the same as for the load2 interface.

logging This contains the Smack logging state.

mapped This contains the CIPSO level used for Smack mapped label representation in network packets.

netlabel This interface allows specific internet addresses to be treated as single label hosts. Packets are sent to single label hosts without CIPSO headers, but only from processes that have Smack write access to the host label. All packets received from single label hosts are given the specified label. The format accepted on write is:

```
"%d.%d.%d.%d label" or "%d.%d.%d.%d/%d label".
```

If the label specified is "-CIPSO" the address is treated as a host that supports CIPSO headers.

onlycap This contains labels processes must have for CAP_MAC_ADMIN and CAP_MAC_OVERRIDE to be effective. If this file is empty these capabilities are effective at for processes with any label. The values are set by writing the desired labels, separated by spaces, to the file or cleared by writing "-" to the file.

ptrace This is used to define the current ptrace policy

- **0 - default:** this is the policy that relies on Smack access rules. For the PTRACE_READ a subject needs to have a read access on object. For the PTRACE_ATTACH a read-write access is required.
- **1 - exact:** this is the policy that limits PTRACE_ATTACH. Attach is only allowed when subject's and object's labels are equal. PTRACE_READ is not affected. Can be overridden with CAP_SYS_PTRACE.
- **2 - draconian:** this policy behaves like the 'exact' above with an exception that it can't be overridden with CAP_SYS_PTRACE.

revoke-subject Writing a Smack label here sets the access to '-' for all access rules with that subject label.

unconfined If the kernel is configured with CONFIG_SECURITY_SMACK_BRINGUP a process with CAP_MAC_ADMIN can write a label into this interface. Thereafter, accesses that involve that label will be logged and the access permitted if it wouldn't be otherwise. Note that this is dangerous and can ruin the proper labeling of your system. It should never be used in production.

relabel-self This interface contains a list of labels to which the process can transition to, by writing to /proc/self/attr/current. Normally a process can change its own label to any legal value, but only if it has CAP_MAC_ADMIN. This interface allows a process without CAP_MAC_ADMIN to relabel itself to one of labels from predefined list. A process without CAP_MAC_ADMIN can change its label only once. When it does, this list will be cleared. The values are set by writing the desired labels, separated by spaces, to the file or cleared by writing "-" to the file.

If you are using the smackload utility you can add access rules in /etc/smack/accesses. They take the form:

```
subjectlabel objectlabel access
```

access is a combination of the letters rwxatb which specify the kind of access permitted a subject with subjectlabel on an object with objectlabel. If there is no rule no access is allowed.

Look for additional programs on http://schaufler-ca.com

The Simplified Mandatory Access Control Kernel (Whitepaper)

Casey Schaufler casey@schaufler-ca.com

Mandatory Access Control

Computer systems employ a variety of schemes to constrain how information is shared among the people and services using the machine. Some of these schemes allow the program or user to decide what other programs or users are allowed access to pieces of data. These schemes are called discretionary access control mechanisms because the access control is specified at the discretion of the user. Other schemes do not leave the decision regarding what a user or program can access up to users or programs. These schemes are called mandatory access control mechanisms because you don't have a choice regarding the users or programs that have access to pieces of data.

Bell & LaPadula

From the middle of the 1980's until the turn of the century Mandatory Access Control (MAC) was very closely associated with the Bell & LaPadula security model, a mathematical description of the United States Department of Defense policy for marking paper documents. MAC in this form enjoyed a following

within the Capital Beltway and Scandinavian supercomputer centers but was often sited as failing to address general needs.

Domain Type Enforcement

Around the turn of the century Domain Type Enforcement (DTE) became popular. This scheme organizes users, programs, and data into domains that are protected from each other. This scheme has been widely deployed as a component of popular Linux distributions. The administrative overhead required to maintain this scheme and the detailed understanding of the whole system necessary to provide a secure domain mapping leads to the scheme being disabled or used in limited ways in the majority of cases.

Smack

Smack is a Mandatory Access Control mechanism designed to provide useful MAC while avoiding the pitfalls of its predecessors. The limitations of Bell & LaPadula are addressed by providing a scheme whereby access can be controlled according to the requirements of the system and its purpose rather than those imposed by an arcane government policy. The complexity of Domain Type Enforcement and avoided by defining access controls in terms of the access modes already in use.

Smack Terminology

The jargon used to talk about Smack will be familiar to those who have dealt with other MAC systems and shouldn't be too difficult for the uninitiated to pick up. There are four terms that are used in a specific way and that are especially important:

Subject: A subject is an active entity on the computer system. On Smack a subject is a task, which is in turn the basic unit of execution.

Object: An object is a passive entity on the computer system. On Smack files of all types, IPC, and tasks can be objects.

Access: Any attempt by a subject to put information into or get information from an object is an access.

Label: Data that identifies the Mandatory Access Control characteristics of a subject or an object.

These definitions are consistent with the traditional use in the security community. There are also some terms from Linux that are likely to crop up:

Capability: A task that possesses a capability has permission to violate an aspect of the system security policy, as identified by the specific capability. A task that possesses one or more capabilities is a privileged task, whereas a task with no capabilities is an unprivileged task.

Privilege: A task that is allowed to violate the system security policy is said to have privilege. As of this writing a task can have privilege either by possessing capabilities or by having an effective user of root.

Smack Basics

Smack is an extension to a Linux system. It enforces additional restrictions on what subjects can access which objects, based on the labels attached to each of the subject and the object.

Labels

Smack labels are ASCII character strings. They can be up to 255 characters long, but keeping them to twenty-three characters is recommended. Single character labels using special characters, that being anything other than a letter or digit, are reserved for use by the Smack development team. Smack

labels are unstructured, case sensitive, and the only operation ever performed on them is comparison for equality. Smack labels cannot contain unprintable characters, the "/" (slash), the "" (backslash), the ""' (quote) and '"' (double-quote) characters. Smack labels cannot begin with a '-'. This is reserved for special options.

There are some predefined labels:

```
_       Pronounced "floor", a single underscore character.
^       Pronounced "hat", a single circumflex character.
*       Pronounced "star", a single asterisk character.
?       Pronounced "huh", a single question mark character.
@       Pronounced "web", a single at sign character.
```

Every task on a Smack system is assigned a label. The Smack label of a process will usually be assigned by the system initialization mechanism.

Access Rules

Smack uses the traditional access modes of Linux. These modes are read, execute, write, and occasionally append. There are a few cases where the access mode may not be obvious. These include:

Signals: A signal is a write operation from the subject task to the object task.

Internet Domain IPC: Transmission of a packet is considered a write operation from the source task to the destination task.

Smack restricts access based on the label attached to a subject and the label attached to the object it is trying to access. The rules enforced are, in order:

1. Any access requested by a task labeled "*" is denied.
2. A read or execute access requested by a task labeled "^" is permitted.
3. A read or execute access requested on an object labeled "_" is permitted.
4. Any access requested on an object labeled "*" is permitted.
5. Any access requested by a task on an object with the same label is permitted.
6. Any access requested that is explicitly defined in the loaded rule set is permitted.
7. Any other access is denied.

Smack Access Rules

With the isolation provided by Smack access separation is simple. There are many interesting cases where limited access by subjects to objects with different labels is desired. One example is the familiar spy model of sensitivity, where a scientist working on a highly classified project would be able to read documents of lower classifications and anything she writes will be "born" highly classified. To accommodate such schemes Smack includes a mechanism for specifying rules allowing access between labels.

Access Rule Format

The format of an access rule is:

```
subject-label object-label access
```

Where subject-label is the Smack label of the task, object-label is the Smack label of the thing being accessed, and access is a string specifying the sort of access allowed. The access specification is searched for letters that describe access modes:

a: indicates that append access should be granted. r: indicates that read access should be granted. w: indicates that write access should be granted. x: indicates that execute access should be granted. t: indicates that the rule requests transmutation. b: indicates that the rule should be reported for bring-up.

Uppercase values for the specification letters are allowed as well. Access mode specifications can be in any order. Examples of acceptable rules are:

```
TopSecret Secret  rx
Secret    Unclass R
Manager   Game    x
User      HR      w
Snap      Crackle rwxatb
New       Old     rRrRr
Closed    Off     -
```

Examples of unacceptable rules are:

```
Top Secret Secret     rx
Ace        Ace        r
Odd        spells     waxbeans
```

Spaces are not allowed in labels. Since a subject always has access to files with the same label specifying a rule for that case is pointless. Only valid letters (rwxatbRWXATB) and the dash ('-') character are allowed in access specifications. The dash is a placeholder, so "a-r" is the same as "ar". A lone dash is used to specify that no access should be allowed.

Applying Access Rules

The developers of Linux rarely define new sorts of things, usually importing schemes and concepts from other systems. Most often, the other systems are variants of Unix. Unix has many endearing properties, but consistency of access control models is not one of them. Smack strives to treat accesses as uniformly as is sensible while keeping with the spirit of the underlying mechanism.

File system objects including files, directories, named pipes, symbolic links, and devices require access permissions that closely match those used by mode bit access. To open a file for reading read access is required on the file. To search a directory requires execute access. Creating a file with write access requires both read and write access on the containing directory. Deleting a file requires read and write access to the file and to the containing directory. It is possible that a user may be able to see that a file exists but not any of its attributes by the circumstance of having read access to the containing directory but not to the differently labeled file. This is an artifact of the file name being data in the directory, not a part of the file.

If a directory is marked as transmuting (SMACK64TRANSMUTE=TRUE) and the access rule that allows a process to create an object in that directory includes 't' access the label assigned to the new object will be that of the directory, not the creating process. This makes it much easier for two processes with different labels to share data without granting access to all of their files.

IPC objects, message queues, semaphore sets, and memory segments exist in flat namespaces and access requests are only required to match the object in question.

Process objects reflect tasks on the system and the Smack label used to access them is the same Smack label that the task would use for its own access attempts. Sending a signal via the kill() system call is a write operation from the signaler to the recipient. Debugging a process requires both reading and writing. Creating a new task is an internal operation that results in two tasks with identical Smack labels and requires no access checks.

Sockets are data structures attached to processes and sending a packet from one process to another requires that the sender have write access to the receiver. The receiver is not required to have read access to the sender.

Setting Access Rules

The configuration file /etc/smack/accesses contains the rules to be set at system startup. The contents are written to the special file /sys/fs/smackfs/load2. Rules can be added at any time and take effect immediately. For any pair of subject and object labels there can be only one rule, with the most recently specified overriding any earlier specification.

Task Attribute

The Smack label of a process can be read from /proc/<pid>/attr/current. A process can read its own Smack label from /proc/self/attr/current. A privileged process can change its own Smack label by writing to /proc/self/attr/current but not the label of another process.

File Attribute

The Smack label of a filesystem object is stored as an extended attribute named SMACK64 on the file. This attribute is in the security namespace. It can only be changed by a process with privilege.

Privilege

A process with CAP_MAC_OVERRIDE or CAP_MAC_ADMIN is privileged. CAP_MAC_OVERRIDE allows the process access to objects it would be denied otherwise. CAP_MAC_ADMIN allows a process to change Smack data, including rules and attributes.

Smack Networking

As mentioned before, Smack enforces access control on network protocol transmissions. Every packet sent by a Smack process is tagged with its Smack label. This is done by adding a CIPSO tag to the header of the IP packet. Each packet received is expected to have a CIPSO tag that identifies the label and if it lacks such a tag the network ambient label is assumed. Before the packet is delivered a check is made to determine that a subject with the label on the packet has write access to the receiving process and if that is not the case the packet is dropped.

CIPSO Configuration

It is normally unnecessary to specify the CIPSO configuration. The default values used by the system handle all internal cases. Smack will compose CIPSO label values to match the Smack labels being used without administrative intervention. Unlabeled packets that come into the system will be given the ambient label.

Smack requires configuration in the case where packets from a system that is not Smack that speaks CIPSO may be encountered. Usually this will be a Trusted Solaris system, but there are other, less widely deployed systems out there. CIPSO provides 3 important values, a Domain Of Interpretation (DOI), a level, and a category set with each packet. The DOI is intended to identify a group of systems that use compatible labeling schemes, and the DOI specified on the Smack system must match that of the remote system or packets will be discarded. The DOI is 3 by default. The value can be read from /sys/fs/smackfs/doi and can be changed by writing to /sys/fs/smackfs/doi.

The label and category set are mapped to a Smack label as defined in /etc/smack/cipso.

A Smack/CIPSO mapping has the form:

```
smack level [category [category]*]
```

Smack does not expect the level or category sets to be related in any particular way and does not assume or assign accesses based on them. Some examples of mappings:

```
TopSecret 7
TS:A,B    7 1 2
SecBDE    5 2 4 6
RAFTERS   7 12 26
```

The ":" and "," characters are permitted in a Smack label but have no special meaning.

The mapping of Smack labels to CIPSO values is defined by writing to /sys/fs/smackfs/cipso2.

In addition to explicit mappings Smack supports direct CIPSO mappings. One CIPSO level is used to indicate that the category set passed in the packet is in fact an encoding of the Smack label. The level used is 250 by default. The value can be read from /sys/fs/smackfs/direct and changed by writing to /sys/fs/smackfs/direct.

Socket Attributes

There are two attributes that are associated with sockets. These attributes can only be set by privileged tasks, but any task can read them for their own sockets.

SMACK64IPIN: The Smack label of the task object. A privileged program that will enforce policy may set this to the star label.

SMACK64IPOUT: The Smack label transmitted with outgoing packets. A privileged program may set this to match the label of another task with which it hopes to communicate.

Smack Netlabel Exceptions

You will often find that your labeled application has to talk to the outside, unlabeled world. To do this there's a special file /sys/fs/smackfs/netlabel where you can add some exceptions in the form of:

```
@IP1       LABEL1 or
@IP2/MASK  LABEL2
```

It means that your application will have unlabeled access to @IP1 if it has write access on LABEL1, and access to the subnet @IP2/MASK if it has write access on LABEL2.

Entries in the /sys/fs/smackfs/netlabel file are matched by longest mask first, like in classless IPv4 routing.

A special label '@' and an option '-CIPSO' can be used there:

```
@      means Internet, any application with any label has access to it
-CIPSO means standard CIPSO networking
```

If you don't know what CIPSO is and don't plan to use it, you can just do:

```
echo 127.0.0.1 -CIPSO > /sys/fs/smackfs/netlabel
echo 0.0.0.0/0 @      > /sys/fs/smackfs/netlabel
```

If you use CIPSO on your 192.168.0.0/16 local network and need also unlabeled Internet access, you can have:

```
echo 127.0.0.1      -CIPSO > /sys/fs/smackfs/netlabel
echo 192.168.0.0/16 -CIPSO > /sys/fs/smackfs/netlabel
echo 0.0.0.0/0      @      > /sys/fs/smackfs/netlabel
```

Writing Applications for Smack

There are three sorts of applications that will run on a Smack system. How an application interacts with Smack will determine what it will have to do to work properly under Smack.

Smack Ignorant Applications

By far the majority of applications have no reason whatever to care about the unique properties of Smack. Since invoking a program has no impact on the Smack label associated with the process the only concern likely to arise is whether the process has execute access to the program.

Smack Relevant Applications

Some programs can be improved by teaching them about Smack, but do not make any security decisions themselves. The utility ls(1) is one example of such a program.

Smack Enforcing Applications

These are special programs that not only know about Smack, but participate in the enforcement of system policy. In most cases these are the programs that set up user sessions. There are also network services that provide information to processes running with various labels.

File System Interfaces

Smack maintains labels on file system objects using extended attributes. The Smack label of a file, directory, or other file system object can be obtained using getxattr(2):

```
len = getxattr("/", "security.SMACK64", value, sizeof (value));
```

will put the Smack label of the root directory into value. A privileged process can set the Smack label of a file system object with setxattr(2):

```
len = strlen("Rubble");
rc = setxattr("/foo", "security.SMACK64", "Rubble", len, 0);
```

will set the Smack label of /foo to “Rubble” if the program has appropriate privilege.

Socket Interfaces

The socket attributes can be read using fgetxattr(2).

A privileged process can set the Smack label of outgoing packets with fsetxattr(2):

```
len = strlen("Rubble");
rc = fsetxattr(fd, "security.SMACK64IPOUT", "Rubble", len, 0);
```

will set the Smack label “Rubble” on packets going out from the socket if the program has appropriate privilege:

```
rc = fsetxattr(fd, "security.SMACK64IPIN, "*", strlen("*"), 0);
```

will set the Smack label “*” as the object label against which incoming packets will be checked if the program has appropriate privilege.

Administration

Smack supports some mount options:

smackfsdef=label: specifies the label to give files that lack the Smack label extended attribute.

smackfsroot=label: specifies the label to assign the root of the file system if it lacks the Smack extended attribute.

smackfshat=label: specifies a label that must have read access to all labels set on the filesystem. Not yet enforced.

smackfsfloor=label: specifies a label to which all labels set on the filesystem must have read access. Not yet enforced.

These mount options apply to all file system types.

Smack auditing

If you want Smack auditing of security events, you need to set CONFIG_AUDIT in your kernel configuration. By default, all denied events will be audited. You can change this behavior by writing a single character to the /sys/fs/smackfs/logging file:

```
0 : no logging
1 : log denied (default)
2 : log accepted
3 : log denied & accepted
```

Events are logged as 'key=value' pairs, for each event you at least will get the subject, the object, the rights requested, the action, the kernel function that triggered the event, plus other pairs depending on the type of event audited.

Bringup Mode

Bringup mode provides logging features that can make application configuration and system bringup easier. Configure the kernel with CONFIG_SECURITY_SMACK_BRINGUP to enable these features. When bringup mode is enabled accesses that succeed due to rules marked with the "b" access mode will logged. When a new label is introduced for processes rules can be added aggressively, marked with the "b". The logging allows tracking of which rules actual get used for that label.

Another feature of bringup mode is the "unconfined" option. Writing a label to /sys/fs/smackfs/unconfined makes subjects with that label able to access any object, and objects with that label accessible to all subjects. Any access that is granted because a label is unconfined is logged. This feature is dangerous, as files and directories may be created in places they couldn't if the policy were being enforced.

TOMOYO

What is TOMOYO?

TOMOYO is a name-based MAC extension (LSM module) for the Linux kernel.

LiveCD-based tutorials are available at

http://tomoyo.sourceforge.jp/1.8/ubuntu12.04-live.html http://tomoyo.sourceforge.jp/1.8/centos6-live.html

Though these tutorials use non-LSM version of TOMOYO, they are useful for you to know what TOMOYO is.

How to enable TOMOYO?

Build the kernel with `CONFIG_SECURITY_TOMOYO=y` and pass `security=tomoyo` on kernel's command line.

Please see http://tomoyo.osdn.jp/2.5/ for details.

Where is documentation?

User <-> Kernel interface documentation is available at http://tomoyo.osdn.jp/2.5/policy-specification/index.html .

Materials we prepared for seminars and symposiums are available at http://osdn.jp/projects/tomoyo/docs/?category_id=532&language_id=1 . Below lists are chosen from three aspects.

What is TOMOYO?

TOMOYO Linux Overview http://osdn.jp/projects/tomoyo/docs/lca2009-takeda.pdf

TOMOYO Linux: pragmatic and manageable security for Linux http://osdn.jp/projects/tomoyo/docs/freedomhectaipei-tomoyo.pdf

TOMOYO Linux: A Practical Method to Understand and Protect Your Own Linux Box http://osdn.jp/projects/tomoyo/docs/PacSec2007-en-no-demo.pdf

What can TOMOYO do?

Deep inside TOMOYO Linux http://osdn.jp/projects/tomoyo/docs/lca2009-kumaneko.pdf

The role of "pathname based access control" in security. http://osdn.jp/projects/tomoyo/docs/lfj2008-bof.pdf

History of TOMOYO?

Realities of Mainlining http://osdn.jp/projects/tomoyo/docs/lfj2008.pdf

What is future plan?

We believe that inode based security and name based security are complementary and both should be used together. But unfortunately, so far, we cannot enable multiple LSM modules at the same time. We feel sorry that you have to give up SELinux/SMACK/AppArmor etc. when you want to use TOMOYO.

We hope that LSM becomes stackable in future. Meanwhile, you can use non-LSM version of TOMOYO, available at http://tomoyo.osdn.jp/1.8/ . LSM version of TOMOYO is a subset of non-LSM version of TOMOYO. We are planning to port non-LSM version's functionalities to LSM versions.

Yama

Yama is a Linux Security Module that collects system-wide DAC security protections that are not handled by the core kernel itself. This is selectable at build-time with `CONFIG_SECURITY_YAMA`, and can be controlled at run-time through sysctls in `/proc/sys/kernel/yama`:

ptrace_scope

As Linux grows in popularity, it will become a larger target for malware. One particularly troubling weakness of the Linux process interfaces is that a single user is able to examine the memory and running state of any of their processes. For example, if one application (e.g. Pidgin) was compromised, it would be possible for an attacker to attach to other running processes (e.g. Firefox, SSH sessions, GPG agent, etc) to extract additional credentials and continue to expand the scope of their attack without resorting to user-assisted phishing.

This is not a theoretical problem. SSH session hijacking (http://www.storm.net.nz/projects/7) and arbitrary code injection (http://c-skills.blogspot.com/2007/05/injectso.html) attacks already exist and remain possible if ptrace is allowed to operate as before. Since ptrace is not commonly used by non-developers and non-admins, system builders should be allowed the option to disable this debugging system.

For a solution, some applications use `prctl(PR_SET_DUMPABLE,...)` to specifically disallow such ptrace attachment (e.g. ssh-agent), but many do not. A more general solution is to only allow ptrace directly from a parent to a child process (i.e. direct "gdb EXE" and "strace EXE" still work), or with `CAP_SYS_PTRACE` (i.e. "gdb –pid=PID", and "strace -p PID" still work as root).

In mode 1, software that has defined application-specific relationships between a debugging process and its inferior (crash handlers, etc), `prctl(PR_SET_PTRACER,pid,...)` can be used. An inferior can declare which other process (and its descendants) are allowed to call `PTRACE_ATTACH` against it. Only one such declared debugging process can exists for each inferior at a time. For example, this is used by KDE, Chromium, and Firefox's crash handlers, and by Wine for allowing only Wine processes to ptrace each other. If a process wishes to entirely disable these ptrace restrictions, it can call `prctl(PR_SET_PTRACER,PR_SET_PTRACER_ANY,...)` so that any otherwise allowed process (even those in external pid namespaces) may attach.

The sysctl settings (writable only with `CAP_SYS_PTRACE`) are:

0 - classic ptrace permissions: a process can `PTRACE_ATTACH` to any other process running under the same uid, as long as it is dumpable (i.e. did not transition uids, start privileged, or have called `prctl(PR_SET_DUMPABLE...)` already). Similarly, `PTRACE_TRACEME` is unchanged.

1 - restricted ptrace: a process must have a predefined relationship with the inferior it wants to call `PTRACE_ATTACH` on. By default, this relationship is that of only its descendants when the above classic criteria is also met. To change the relationship, an inferior can call `prctl(PR_SET_PTRACER,debugger,...)` to declare an allowed debugger PID to call `PTRACE_ATTACH` on the inferior. Using `PTRACE_TRACEME` is unchanged.

2 - admin-only attach: only processes with `CAP_SYS_PTRACE` may use ptrace with `PTRACE_ATTACH`, or through children calling `PTRACE_TRACEME`.

3 - no attach: no processes may use ptrace with `PTRACE_ATTACH` nor via `PTRACE_TRACEME`. Once set, this sysctl value cannot be changed.

The original children-only logic was based on the restrictions in grsecurity.