NAME

bootparam - introduction to boot time parameters of the Linux kernel

DESCRIPTION

The Linux kernel accepts certain 'command-line options' or 'boot time parameters' at the moment it is started. In general this is used to supply the kernel with information about hardware parameters that the kernel would not be able to determine on its own, or to avoid/override the values that the kernel would otherwise detect.

When the kernel is booted directly by the BIOS (say from a floppy to which you copied a kernel using 'cp zImage /dev/fd0'), you have no opportunity to specify any parameters. So, in order to take advantage of this possibility you have to use a boot loader that is able to pass parameters, such as GRUB.

The argument list

The kernel command line is parsed into a list of strings (boot arguments) separated by spaces. Most of the boot arguments take have the form:

name[=value_1][,value_2]...[,value_10]

where 'name' is a unique keyword that is used to identify what part of the kernel the associated values (if any) are to be given to. Note the limit of 10 is real, as the present code handles only 10 comma separated parameters per keyword. (However, you can reuse the same keyword with up to an additional 10 parameters in unusually complicated situations, assuming the setup function supports it.)

Most of the sorting is coded in the kernel source file *init/main.c*. First, the kernel checks to see if the argument is any of the special arguments 'root=', 'nfsroot=', 'nfsaddrs=', 'ro', 'rw', 'debug' or 'init'. The meaning of these special arguments is described below.

Then it walks a list of setup functions (contained in the bootsetups array) to see if the specified argument string (such as 'foo') has been associated with a setup function ('foo_setup()') for a particular device or part of the kernel. If you passed the kernel the line foo=3,4,5,6 then the kernel would search the bootsetups array to see if 'foo' was registered. If it was, then it would call the setup function associated with 'foo' (foo_setup()) and hand it the arguments 3, 4, 5, and 6 as given on the kernel command line.

Anything of the form 'foo=bar' that is not accepted as a setup function as described above is then interpreted as an environment variable to be set. A (useless?) example would be to use 'TERM=vt100' as a boot argument.

Any remaining arguments that were not picked up by the kernel and were not interpreted as environment variables are then passed onto process one, which is usually the init(1) program. The most common argument that is passed to the *init* process is the word 'single' which instructs it to boot the computer in single user mode, and not launch all the usual daemons. Check the manual page for the version of init(1) installed on your system to see what arguments it accepts.

General non-device-specific boot arguments

'init=...'

This sets the initial command to be executed by the kernel. If this is not set, or cannot be found, the kernel will try */sbin/init*, then */etc/init*, then */bin/init*, then */bin/sh* and panic if all of this fails.

'nfsaddrs=...'

This sets the nfs boot address to the given string. This boot address is used in case of a net boot.

'nfsroot=...'

This sets the nfs root name to the given string. If this string does not begin with '/' or ',' or a digit, then it is prefixed by '/tftpboot/'. This root name is used in case of a net boot.

'no387'

(Only when **CONFIG_BUGi386** is defined.) Some i387 coprocessor chips have bugs that show up when used in 32 bit protected mode. For example, some of the early ULSI-387 chips would cause solid lockups while performing floating-point calculations. Using the 'no387' boot argument causes Linux to ignore the maths coprocessor even if you have one. Of course you must then

have your kernel compiled with math emulation support!

'no-hlt'

(Only when **CONFIG_BUGi386** is defined.) Some of the early i486DX-100 chips have a problem with the 'hlt' instruction, in that they can't reliably return to operating mode after this instruction is used. Using the 'no-hlt' instruction tells Linux to just run an infinite loop when there is nothing else to do, and to not halt the CPU. This allows people with these broken chips to use Linux.

'root=...'

This argument tells the kernel what device is to be used as the root filesystem while booting. The default of this setting is determined at compile time, and usually is the value of the root device of the system that the kernel was built on. To override this value, and select the second floppy drive as the root device, one would use 'root=/dev/fd1'.

The root device can be specified symbolically or numerically. A symbolic specification has the form */dev/XXYN*, where XX designates the device type ('hd' for ST-506 compatible hard disk, with Y in 'a'-'d'; 'sd' for SCSI compatible disk, with Y in 'a'-'e'; 'ad' for Atari ACSI disk, with Y in 'a'-'e', 'ez' for a Syquest EZ135 parallel port removable drive, with Y='a', 'xd' for XT compatible disk, with Y either 'a' or 'b'; 'fd' for floppy disk, with Y the floppy drive number—fd0 would be the DOS 'A:' drive, and fd1 would be 'B:'), Y the driver letter or number, and N the number (in decimal) of the partition on this device (absent in the case of floppies). Recent kernels allow many other types, mostly for CD-ROMs: nfs, ram, scd, mcd, cdu535, aztcd, cm206cd, gscd, sbpcd, sonycd, bpcd. (The type nfs specifies a net boot; ram refers to a ram disk.)

Note that this has nothing to do with the designation of these devices on your filesystem. The '/dev/' part is purely conventional.

The more awkward and less portable numeric specification of the above possible root devices in major/minor format is also accepted. (For example, */dev/sda3* is major 8, minor 3, so you could use 'root=0x803' as an alternative.)

'rootdelay='

This parameter sets the delay (in seconds) to pause before attempting to mount the root filesystem.

'rootflags=...'

This parameter sets the mount option string for the root filesystem (see also fstab(5)).

'rootfstype=...'

The 'rootfstype' option tells the kernel to mount the root filesystem as if it where of the type specified. This can be useful (for example) to mount an ext3 filesystem as ext2 and then remove the journal in the root filesystem, in fact reverting its format from ext3 to ext2 without the need to boot the box from alternate media.

'ro' and **'rw'**

The 'ro' option tells the kernel to mount the root filesystem as 'read-only' so that filesystem consistency check programs (fsck) can do their work on a quiescent filesystem. No processes can write to files on the filesystem in question until it is 'remounted' as read/write capable, for example, by 'mount -w -n -o remount /'. (See also mount(8).)

The 'rw' option tells the kernel to mount the root filesystem read/write. This is the default.

'resume=...'

This tells the kernel the location of the suspend-to-disk data that you want the machine to resume from after hibernation. Usually, it is the same as your swap partition or file. Example:

resume=/dev/hda2

'reserve=...'

This is used to protect I/O port regions from probes. The form of the command is:

reserve=*iobase,extent[,iobase,extent]...*

In some machines it may be necessary to prevent device drivers from checking for devices (autoprobing) in a specific region. This may be because of hardware that reacts badly to the probing, or hardware that would be mistakenly identified, or merely hardware you don't want the kernel to initialize.

The reserve boot-time argument specifies an I/O port region that shouldn't be probed. A device driver will not probe a reserved region, unless another boot argument explicitly specifies that it do so.

For example, the boot line

reserve=0x300,32 blah=0x300

keeps all device drivers except the driver for 'blah' from probing 0x300-0x31f.

'mem=...'

The BIOS call defined in the PC specification that returns the amount of installed memory was designed only to be able to report up to 64MB. Linux uses this BIOS call at boot to determine how much memory is installed. If you have more than 64MB of RAM installed, you can use this boot argument to tell Linux how much memory you have. The value is in decimal or hexadecimal (prefix 0x), and the suffixes 'k' (times 1024) or 'M' (times 1048576) can be used. Here is a quote from Linus on usage of the 'mem=' parameter.

> The kernel will accept any 'mem=xx' parameter you give it, and if it turns out that you lied to it, it will crash horribly sooner or later. The parameter indicates the highest addressable RAM address, so 'mem=0x1000000' means you have 16MB of memory, for example. For a 96MB machine this would be 'mem=0x6000000'.
>
> **NOTE**: some machines might use the top of memory for BIOS caching or whatever, so you might not actually have up to the full 96MB addressable. The reverse is also true: some chipsets will map the physical memory that is covered by the BIOS area into the area just past the top of memory, so the top-of-mem might actually be 96MB + 384kB for example. If you tell linux that it has more memory than it actually does have, bad things will happen: maybe not at once, but surely eventually.

You can also use the boot argument 'mem=nopentium' to turn off 4 MB page tables on kernels configured for IA32 systems with a pentium or newer CPU.

'panic=N'

By default the kernel will not reboot after a panic, but this option will cause a kernel reboot after N seconds (if N is greater than zero). This panic timeout can also be set by

echo N > /proc/sys/kernel/panic

'reboot=[warm|cold][,[bios|hard]]'

(Only when **CONFIG_BUGi386** is defined.) Since 2.0.22 a reboot is by default a cold reboot. One asks for the old default with 'reboot=warm'. (A cold reboot may be required to reset certain hardware, but might destroy not yet written data in a disk cache. A warm reboot may be faster.) By default a reboot is hard, by asking the keyboard controller to pulse the reset line low, but there is at least one type of motherboard where that doesn't work. The option 'reboot=bios' will instead jump through the BIOS.

'nosmp' and **'maxcpus=N'**

(Only when __SMP__ is defined.) A command-line option of 'nosmp' or 'maxcpus=0' will disable SMP activation entirely; an option 'maxcpus=N' limits the maximum number of CPUs activated in SMP mode to N.

Boot arguments for use by kernel developers

'debug'

Kernel messages are handed off to the kernel log daemon klogd so that they may be logged to disk. Messages with a priority above *console_loglevel* are also printed on the console. (For these levels, see <*linux/kernel.h*>.) By default this variable is set to log anything more important than

debug messages. This boot argument will cause the kernel to also print the messages of DEBUG priority. The console loglevel can also be set at run time via an option to klogd. See **klogd(8).**

'profile=N'

It is possible to enable a kernel profiling function, if one wishes to find out where the kernel is spending its CPU cycles. Profiling is enabled by setting the variable *prof_shift* to a nonzero value. This is done either by specifying **CONFIG_PROFILE** at compile time, or by giving the 'profile=' option. Now the value that *prof_shift* gets will be N, when given, or **CONFIG_PROFILE_SHIFT**, when that is given, or 2, the default. The significance of this variable is that it gives the granularity of the profiling: each clock tick, if the system was executing kernel code, a counter is incremented:

```
profile[address >> prof_shift]++;
```

The raw profiling information can be read from */proc/profile*. Probably you'll want to use a tool such as readprofile.c to digest it. Writing to */proc/profile* will clear the counters.

'swap=N1,N2,N3,N4,N5,N6,N7,N8'

Set the eight parameters max_page_age, page_advance, page_decline, page_initial_age, age_cluster_fract, age_cluster_min, pageout_weight, bufferout_weight that control the kernel swap algorithm. For kernel tuners only.

'buff=N1,N2,N3,N4,N5,N6'

Set the six parameters max_buff_age, buff_advance, buff_decline, buff_initial_age, bufferout_weight, buffermem_grace that control kernel buffer memory management. For kernel tuners only.

Boot arguments for ramdisk use

(Only if the kernel was compiled with **CONFIG_BLK_DEV_RAM**.) In general it is a bad idea to use a ramdisk under Linux—the system will use available memory more efficiently itself. But while booting (or while constructing boot floppies) it is often useful to load the floppy contents into a ramdisk. One might also have a system in which first some modules (for filesystem or hardware) must be loaded before the main disk can be accessed.

In Linux 1.3.48, ramdisk handling was changed drastically. Earlier, the memory was allocated statically, and there was a 'ramdisk=N' parameter to tell its size. (This could also be set in the kernel image at compile time.) These days ram disks use the buffer cache, and grow dynamically. For a lot of information in conjunction with the new ramdisk setup, see the kernel source file *Documentation/blockdev/ramdisk.txt* (*Documentation/ramdisk.txt* in older kernels).

There are four parameters, two boolean and two integral.

'load_ramdisk=N'

If N=1, do load a ramdisk. If N=0, do not load a ramdisk. (This is the default.)

'prompt_ramdisk=N'

If N=1, do prompt for insertion of the floppy. (This is the default.) If N=0, do not prompt. (Thus, this parameter is never needed.)

'ramdisk_size=N' or (obsolete) **'ramdisk=N'**

Set the maximal size of the ramdisk(s) to N kB. The default is 4096 (4 MB).

'ramdisk_start=N'

Sets the starting block number (the offset on the floppy where the ramdisk starts) to N. This is needed in case the ramdisk follows a kernel image.

'noinitrd'

(Only if the kernel was compiled with **CONFIG_BLK_DEV_RAM** and **CONFIG_BLK_DEV_INITRD**.) These days it is possible to compile the kernel to use initrd. When this feature is enabled, the boot process will load the kernel and an initial ramdisk; then the kernel converts initrd into a "normal" ramdisk, which is mounted read-write as root device; then */linuxrc* is executed; afterward the "real" root filesystem is mounted, and the initrd filesystem is moved

over to */initrd*; finally the usual boot sequence (e.g., invocation of */sbin/init*) is performed.

For a detailed description of the initrd feature, see the kernel source file *Documentation/initrd.txt*.

The 'noinitrd' option tells the kernel that although it was compiled for operation with initrd, it should not go through the above steps, but leave the initrd data under */dev/initrd*. (This device can be used only once: the data is freed as soon as the last process that used it has closed */dev/initrd*.)

Boot arguments for SCSI devices

General notation for this section:

iobase -- the first I/O port that the SCSI host occupies. These are specified in hexadecimal notation, and usually lie in the range from 0x200 to 0x3ff.

irq -- the hardware interrupt that the card is configured to use. Valid values will be dependent on the card in question, but will usually be 5, 7, 9, 10, 11, 12, and 15. The other values are usually used for common peripherals like IDE hard disks, floppies, serial ports, and so on.

scsi-id -- the ID that the host adapter uses to identify itself on the SCSI bus. Only some host adapters allow you to change this value, as most have it permanently specified internally. The usual default value is 7, but the Seagate and Future Domain TMC-950 boards use 6.

parity -- whether the SCSI host adapter expects the attached devices to supply a parity value with all information exchanges. Specifying a one indicates parity checking is enabled, and a zero disables parity checking. Again, not all adapters will support selection of parity behavior as a boot argument.

'max_scsi_luns=...'

A SCSI device can have a number of 'subdevices' contained within itself. The most common example is one of the new SCSI CD-ROMs that handle more than one disk at a time. Each CD is addressed as a 'Logical Unit Number' (LUN) of that particular device. But most devices, such as hard disks, tape drives and such are only one device, and will be assigned to LUN zero.

Some poorly designed SCSI devices cannot handle being probed for LUNs not equal to zero. Therefore, if the compile-time flag **CONFIG_SCSI_MULTI_LUN** is not set, newer kernels will by default only probe LUN zero.

To specify the number of probed LUNs at boot, one enters 'max_scsi_luns=n' as a boot arg, where n is a number between one and eight. To avoid problems as described above, one would use n=1 to avoid upsetting such broken devices.

SCSI tape configuration

Some boot time configuration of the SCSI tape driver can be achieved by using the following:

st=*buf_size[,write_threshold[,max_bufs]]*

The first two numbers are specified in units of kB. The default *buf_size* is 32kB, and the maximum size that can be specified is a ridiculous 16384kB. The *write_threshold* is the value at which the buffer is committed to tape, with a default value of 30kB. The maximum number of buffers varies with the number of drives detected, and has a default of two. An example usage would be:

st=32,30,2

Full details can be found in the file *Documentation/scsi/st.txt* (or *drivers/scsi/README.st* for older kernels) in the Linux kernel source.

Adaptec aha151x, aha152x, aic6260, aic6360, SB16-SCSI configuration

The aha numbers refer to cards and the aic numbers refer to the actual SCSI chip on these type of cards, including the Soundblaster-16 SCSI.

The probe code for these SCSI hosts looks for an installed BIOS, and if none is present, the probe will not find your card. Then you will have to use a boot argument of the form:

aha152x=*iobase[,irq[,scsi-id[,reconnect[,parity]]]]*

If the driver was compiled with debugging enabled, a sixth value can be specified to set the debug level.

All the parameters are as described at the top of this section, and the *reconnect* value will allow device disconnect/reconnect if a nonzero value is used. An example usage is as follows:

aha152x=0x340,11,7,1

Note that the parameters must be specified in order, meaning that if you want to specify a parity setting, then you will have to specify an iobase, irq, scsi-id and reconnect value as well.

Adaptec aha154x configuration

The aha1542 series cards have an i82077 floppy controller onboard, while the aha1540 series cards do not. These are busmastering cards, and have parameters to set the "fairness" that is used to share the bus with other devices. The boot argument looks like the following.

aha1542=*iobase[,buson,busoff[,dmaspeed]]*

Valid iobase values are usually one of: 0x130, 0x134, 0x230, 0x234, 0x330, 0x334. Clone cards may permit other values.

The *buson*, *busoff* values refer to the number of microseconds that the card dominates the ISA bus. The defaults are 11us on, and 4us off, so that other cards (such as an ISA LANCE Ethernet card) have a chance to get access to the ISA bus.

The *dmaspeed* value refers to the rate (in MB/s) at which the DMA (Direct Memory Access) transfers proceed. The default is 5MB/s. Newer revision cards allow you to select this value as part of the soft-configuration, older cards use jumpers. You can use values up to 10MB/s assuming that your motherboard is capable of handling it. Experiment with caution if using values over 5MB/s.

Adaptec aha274x, aha284x, aic7xxx configuration

These boards can accept an argument of the form:

aic7xxx=*extended,no_reset*

The *extended* value, if nonzero, indicates that extended translation for large disks is enabled. The *no_reset* value, if nonzero, tells the driver not to reset the SCSI bus when setting up the host adapter at boot.

AdvanSys SCSI Hosts configuration ('advansys=')

The AdvanSys driver can accept up to four I/O addresses that will be probed for an AdvanSys SCSI card. Note that these values (if used) do not effect EISA or PCI probing in any way. They are used only for probing ISA and VLB cards. In addition, if the driver has been compiled with debugging enabled, the level of debugging output can be set by adding an 0xdeb[0-f] parameter. The 0-f allows setting the level of the debugging messages to any of 16 levels of verbosity.

AM53C974

Syntax:

AM53C974=*host-scsi-id,target-scsi-id,max-rate,max-offset*

BusLogic SCSI Hosts configuration ('BusLogic=')

Syntax:

BusLogic=*N1,N2,N3,N4,N5,S1,S2,...*

For an extensive discussion of the BusLogic command line parameters, see the kernel source file *drivers/scsi/BusLogic.c*. The text below is a very much abbreviated extract.

The parameters N1-N5 are integers. The parameters S1,... are strings. N1 is the I/O Address at which the Host Adapter is located. N2 is the Tagged Queue Depth to use for Target Devices that support Tagged Queuing. N3 is the Bus Settle Time in seconds. This is the amount of time to wait between a Host Adapter Hard Reset which initiates a SCSI Bus Reset and issuing any SCSI

Commands. N4 is the Local Options (for one Host Adapter). N5 is the Global Options (for all Host Adapters).

The string options are used to provide control over Tagged Queuing (TQ:Default, TQ:Enable, TQ:Disable, TQ:<Per-Target-Spec>), over Error Recovery (ER:Default, ER:HardReset, ER:BusDeviceReset, ER:None, ER:<Per-Target-Spec>), and over Host Adapter Probing (NoProbe, NoProbeISA, NoSortPCI).

EATA/DMA configuration

The default list of I/O ports to be probed can be changed by

eata=*iobase,iobase,...*

Future Domain TMC-16x0 configuration

Syntax:

fdomain=*iobase,irq[,adapter_id]*

Great Valley Products (GVP) SCSI controller configuration

Syntax:

gvp11=*dma_transfer_bitmask*

Future Domain TMC-8xx, TMC-950 configuration

Syntax:

tmc8xx=*mem_base,irq*

The *mem_base* value is the value of the memory-mapped I/O region that the card uses. This will usually be one of the following values: 0xc8000, 0xca000, 0xcc000, 0xce000, 0xdc000, 0xde000.

IN2000 configuration

Syntax:

in2000=*S*

where S is a comma-separated string of items keyword[:value]. Recognized keywords (possibly with value) are: ioport:addr, noreset, nosync:x, period:ns, disconnect:x, debug:x, proc:x. For the function of these parameters, see the kernel source file *drivers/scsi/in2000.c*.

NCR5380 and NCR53C400 configuration

The boot argument is of the form

ncr5380=*iobase,irq,dma*

or

ncr53c400=*iobase,irq*

If the card doesn't use interrupts, then an IRQ value of 255 (0xff) will disable interrupts. An IRQ value of 254 means to autoprobe. More details can be found in the file *Documentation/scsi/g_NCR5380.txt* (or *drivers/scsi/README.g_NCR5380* for older kernels) in the Linux kernel source.

NCR53C8xx configuration

Syntax:

ncr53c8xx=*S*

where S is a comma-separated string of items keyword:value. Recognized keywords are: mpar (master_parity), spar (scsi_parity), disc (disconnection), specf (special_features), ultra (ultra_scsi), fsn (force_sync_nego), tags (default_tags), sync (default_sync), verb (verbose), debug (debug), burst (burst_max). For the function of the assigned values, see the kernel source file *drivers/scsi/ncr53c8xx.c*.

NCR53c406a configuration

Syntax:

ncr53c406a=*iobase[,irq[,fastpio]]*

Specify irq = 0 for noninterrupt driven mode. Set fastpio = 1 for fast pio mode, 0 for slow mode.

Pro Audio Spectrum configuration

The PAS16 uses a NC5380 SCSI chip, and newer models support jumperless configuration. The boot argument is of the form:

pas16=*iobase,irq*

The only difference is that you can specify an IRQ value of 255, which will tell the driver to work without using interrupts, albeit at a performance loss. The iobase is usually 0x388.

Seagate ST-0x configuration

If your card is not detected at boot time, you will then have to use a boot argument of the form:

st0x=*mem_base,irq*

The *mem_base* value is the value of the memory-mapped I/O region that the card uses. This will usually be one of the following values: 0xc8000, 0xca000, 0xcc000, 0xce000, 0xdc000, 0xde000.

Trantor T128 configuration

These cards are also based on the NCR5380 chip, and accept the following options:

t128=*mem_base,irq*

The valid values for *mem_base* are as follows: 0xcc000, 0xc8000, 0xdc000, 0xd8000.

UltraStor 14F/34F configuration

The default list of I/O ports to be probed can be changed by

eata=*iobase,iobase,...*.

WD7000 configuration

Syntax:

wd7000=*irq,dma,iobase*

Commodore Amiga A2091/590 SCSI controller configuration

Syntax:

wd33c93=*S*

where S is a comma-separated string of options. Recognized options are nosync:bitmask, nodma:x, period:ns, disconnect:x, debug:x, clock:x, next. For details, see the kernel source file *drivers/scsi/wd33c93.c*.

Hard disks

IDE Disk/CD-ROM Driver Parameters

The IDE driver accepts a number of parameters, which range from disk geometry specifications, to support for broken controller chips. Drive-specific options are specified by using 'hdX=' with X in 'a'-'h'.

Non-drive-specific options are specified with the prefix 'hd='. Note that using a drive-specific prefix for a non-drive-specific option will still work, and the option will just be applied as expected.

Also note that 'hd=' can be used to refer to the next unspecified drive in the (a, ..., h) sequence. For the following discussions, the 'hd=' option will be cited for brevity. See the file*Documentation/ide.txt* (or *drivers/block/README.ide* for older kernels) in the Linux kernel source for more details.

The 'hd=cyls,heads,sects[,wpcom[,irq]]' options

These options are used to specify the physical geometry of the disk. Only the first three values are required. The cylinder/head/sectors values will be those used by fdisk. The write

precompensation value is ignored for IDE disks. The IRQ value specified will be the IRQ used for the interface that the drive resides on, and is not really a drive-specific parameter.

The 'hd=serialize' option

The dual IDE interface CMD-640 chip is broken as designed such that when drives on the secondary interface are used at the same time as drives on the primary interface, it will corrupt your data. Using this option tells the driver to make sure that both interfaces are never used at the same time.

The 'hd=dtc2278' option

This option tells the driver that you have a DTC-2278D IDE interface. The driver then tries to do DTC-specific operations to enable the second interface and to enable faster transfer modes.

The 'hd=noprobe' option

Do not probe for this drive. For example,

hdb=noprobe hdb=1166,7,17

would disable the probe, but still specify the drive geometry so that it would be registered as a valid block device, and hence usable.

The 'hd=nowerr' option

Some drives apparently have the **WRERR_STAT** bit stuck on permanently. This enables a workaround for these broken devices.

The 'hd=cdrom' option

This tells the IDE driver that there is an ATAPI compatible CD-ROM attached in place of a normal IDE hard disk. In most cases the CD-ROM is identified automatically, but if it isn't then this may help.

Standard ST-506 Disk Driver Options ('hd=')

The standard disk driver can accept geometry arguments for the disks similar to the IDE driver. Note however that it expects only three values (C/H/S); any more or any less and it will silently ignore you. Also, it accepts only 'hd=' as an argument, that is, 'hda=' and so on are not valid here. The format is as follows:

hd=cyls,heads,sects

If there are two disks installed, the above is repeated with the geometry parameters of the second disk.

XT Disk Driver Options ('xd=')

If you are unfortunate enough to be using one of these old 8-bit cards that move data at a whopping 125kB/s, then here is the scoop. If the card is not recognized, you will have to use a boot argument of the form:

xd=type,irq,iobase,dma_chan

The type value specifies the particular manufacturer of the card, overriding autodetection. For the types to use, consult the *drivers/block/xd.c* source file of the kernel you are using. The type is an index in the list *xd_sigs* and in the course of time types have been added to or deleted from the middle of the list, changing all type numbers. Today (Linux 2.5.0) the types are 0=generic; 1=DTC 5150cx; 2,3=DTC 5150x; 4,5=Western Digital; 6,7,8=Seagate; 9=Omti; 10=XEBEC, and where here several types are given with the same designation, they are equivalent.

The xd_setup() function does no checking on the values, and assumes that you entered all four values. Don't disappoint it. Here is an example usage for a WD1002 controller with the BIOS disabled/removed, using the 'default' XT controller parameters:

xd=2,5,0x320,3

Syquest's EZ* removable disks

Syntax:

ez=*iobase[,irq[,rep[,nybble]]]*

IBM MCA bus devices

See also the kernel source file *Documentation/mca.txt*.

PS/2 ESDI hard disks

It is possible to specify the desired geometry at boot time:

ed=*cyls,heads,sectors.*

For a ThinkPad-720, add the option

tp720=1.

IBM Microchannel SCSI Subsystem configuration

Syntax:

ibmmcascsi=*N*

where N is the *pun* (SCSI ID) of the subsystem.

The Aztech Interface

The syntax for this type of card is:

aztcd=iobase[,magic_number]

If you set the magic_number to 0x79, then the driver will try and run anyway in the event of an unknown firmware version. All other values are ignored.

Parallel port CD-ROM drives

Syntax:

pcd.driveN=prt,pro,uni,mod,slv,dly
pcd.nice=nice

where 'port' is the base address, 'pro' is the protocol number, 'uni' is the unit selector (for chained devices), 'mod' is the mode (or -1 to choose the best automatically), 'slv' is 1 if it should be a slave, and 'dly' is a small integer for slowing down port accesses. The 'nice' parameter controls the driver's use of idle CPU time, at the expense of some speed.

The CDU-31A and CDU-33A Sony Interface

This CD-ROM interface is found on some of the Pro Audio Spectrum sound cards, and other Sony supplied interface cards. The syntax is as follows:

cdu31a=iobase,[irq[,is_pas_card]]

Specifying an IRQ value of zero tells the driver that hardware interrupts aren't supported (as on some PAS cards). If your card supports interrupts, you should use them as it cuts down on the CPU usage of the driver.

The *is_pas_card* should be entered as 'PAS' if using a Pro Audio Spectrum card, and otherwise it should not be specified at all.

The CDU-535 Sony Interface

The syntax for this CD-ROM interface is:

sonycd535=iobase[,irq]

A zero can be used for the I/O base as a 'placeholder' if one wishes to specify an IRQ value.

The GoldStar Interface

The syntax for this CD-ROM interface is:

gscd=iobase

The ISP16 CD-ROM Interface

Syntax:

isp16=[iobase[,irq[,dma[,type]]]]

(Three integers and a string.) If the type is given as 'noisp16', the interface will not be configured. Other recognized types are: 'Sanyo", 'Sony', 'Panasonic' and 'Mitsumi'.

The Mitsumi Standard Interface

The syntax for this CD-ROM interface is:

mcd=iobase,[irq[,wait_value]]

The *wait_value* is used as an internal timeout value for people who are having problems with their drive, and may or may not be implemented depending on a compile-time #define. The Mitsumi FX400 is an IDE/ATAPI CD-ROM player and does not use the mcd driver.

The Mitsumi XA/MultiSession Interface

This is for the same hardware as above, but the driver has extended features. Syntax:

mcdx=iobase[,irq]

The Optics Storage Interface

The syntax for this type of card is:

optcd=iobase

The Phillips CM206 Interface

The syntax for this type of card is:

cm206=[iobase][,irq]

The driver assumes numbers between 3 and 11 are IRQ values, and numbers between 0x300 and 0x370 are I/O ports, so you can specify one, or both numbers, in any order. It also accepts 'cm206=auto' to enable autoprobing.

The Sanyo Interface

The syntax for this type of card is:

sjcd=iobase[,irq[,dma_channel]]

The SoundBlaster Pro Interface

The syntax for this type of card is:

sbpcd=iobase,type

where type is one of the following (case sensitive) strings: 'SoundBlaster', 'LaserMate', or 'SPEA'. The I/O base is that of the CD-ROM interface, and not that of the sound portion of the card.

Ethernet devices

Different drivers make use of different parameters, but they all at least share having an IRQ, an I/O port base value, and a name. In its most generic form, it looks something like this:

ether=irq,iobase[,param_1[,...param_8]],name

The first nonnumeric argument is taken as the name. The param_n values (if applicable) usually have different meanings for each different card/driver. Typical param_n values are used to specify things like shared memory address, interface selection, DMA channel and the like.

The most common use of this parameter is to force probing for a second ethercard, as the default is to probe only for one. This can be accomplished with a simple:

ether=0,0,eth1

Note that the values of zero for the IRQ and I/O base in the above example tell the driver(s) to autoprobe.

The Ethernet-HowTo has extensive documentation on using multiple cards and on the card/driver-specific implementation of the param_n values where used. Interested readers should refer to the section in that document on their particular card.

The floppy disk driver

There are many floppy driver options, and they are all listed in *Documentation/floppy.txt* (or *drivers/block/README.fd* for older kernels) in the Linux kernel source. This information is taken directly from that file.

floppy=mask,allowed_drive_mask
Sets the bit mask of allowed drives to mask. By default, only units 0 and 1 of each floppy controller are allowed. This is done because certain nonstandard hardware (ASUS PCI motherboards) mess up the keyboard when accessing units 2 or 3. This option is somewhat obsoleted by the cmos option.

floppy=all_drives
Sets the bit mask of allowed drives to all drives. Use this if you have more than two drives connected to a floppy controller.

floppy=asus_pci
Sets the bit mask to allow only units 0 and 1. (The default)

floppy=daring
Tells the floppy driver that you have a well behaved floppy controller. This allows more efficient and smoother operation, but may fail on certain controllers. This may speed up certain operations.

floppy=0,daring
Tells the floppy driver that your floppy controller should be used with caution.

floppy=one_fdc
Tells the floppy driver that you have only floppy controller (default)

floppy=two_fdc or **floppy=address,two_fdc**
Tells the floppy driver that you have two floppy controllers. The second floppy controller is assumed to be at address. If address is not given, 0x370 is assumed.

floppy=thinkpad
Tells the floppy driver that you have a Thinkpad. Thinkpads use an inverted convention for the disk change line.

floppy=0,thinkpad
Tells the floppy driver that you don't have a Thinkpad.

floppy=drive,type,cmos
Sets the cmos type of drive to type. Additionally, this drive is allowed in the bit mask. This is useful if you have more than two floppy drives (only two can be described in the physical cmos), or if your BIOS uses nonstandard CMOS types. Setting the CMOS to 0 for the first two drives (default) makes the floppy driver read the physical cmos for those drives.

floppy=unexpected_interrupts
Print a warning message when an unexpected interrupt is received (default behavior)

floppy=no_unexpected_interrupts or **floppy=L40SX**
Don't print a message when an unexpected interrupt is received. This is needed on IBM L40SX laptops in certain video modes. (There seems to be an interaction between video and floppy. The unexpected interrupts only affect performance, and can safely be ignored.)

The sound driver

The sound driver can also accept boot arguments to override the compiled in values. This is not recommended, as it is rather complex. It is described in the Linux kernel source file *Documentation/sound/oss/README.OSS* (*drivers/sound/Readme.linux* in older kernel versions). It accepts a boot argument of the form:

sound=device1[,device2[,device3...[,device10]]]

where each deviceN value is of the following format 0xTaaaId and the bytes are used as follows:

T - device type: 1=FM, 2=SB, 3=PAS, 4=GUS, 5=MPU401, 6=SB16, 7=SB16-MPU401

aaa - I/O address in hex.

I - interrupt line in hex (i.e 10=a, 11=b, ...)

d - DMA channel.

As you can see it gets pretty messy, and you are better off to compile in your own personal values as recommended. Using a boot argument of 'sound=0' will disable the sound driver entirely.

ISDN drivers

The ICN ISDN driver

Syntax:

icn=iobase,membase,icn_id1,icn_id2

where icn_id1,icn_id2 are two strings used to identify the card in kernel messages.

The PCBIT ISDN driver

Syntax:

pcbit=membase1,irq1[,membase2,irq2]

where membaseN is the shared memory base of the N'th card, and irqN is the interrupt setting of the N'th card. The default is IRQ 5 and membase 0xD0000.

The Teles ISDN driver

Syntax:

teles=iobase,irq,membase,protocol,teles_id

where iobase is the I/O port address of the card, membase is the shared memory base address of the card, irq is the interrupt channel the card uses, and teles_id is the unique ASCII string identifier.

Serial port drivers

The RISCom/8 Multiport Serial Driver ('riscom8=')

Syntax:

riscom=iobase1[,iobase2[,iobase3[,iobase4]]]

More details can be found in the kernel source file *Documentation/riscom8.txt*.

The DigiBoard Driver ('digi=')

If this option is used, it should have precisely six parameters. Syntax:

digi=status,type,altpin,numports,iobase,membase

The parameters maybe given as integers, or as strings. If strings are used, then iobase and membase should be given in hexadecimal. The integer arguments (fewer may be given) are in order: status (Enable(1) or Disable(0) this card), type (PC/Xi(0), PC/Xe(1), PC/Xeve(2), PC/Xem(3)), altpin (Enable(1) or Disable(0) alternate pin arrangement), numports (number of ports on this card), iobase (I/O Port where card is configured (in HEX)), membase (base of memory window (in HEX)). Thus, the following two boot prompt arguments are equivalent:

digi=E,PC/Xi,D,16,200,D0000
digi=1,0,0,16,0x200,851968

More details can be found in the kernel source file *Documentation/digiboard.txt*.

The Baycom Serial/Parallel Radio Modem

Syntax:

baycom=iobase,irq,modem

There are precisely 3 parameters; for several cards, give several 'baycom=' commands. The modem parameter is a string that can take one of the values ser12, ser12*, par96, par96*. Here the * denotes that software DCD is to be used, and ser12/par96 chooses between the supported

modem types. For more details, see the file *Documentation/networking/baycom.txt* (or *drivers/net/README.baycom* for older kernels) in the Linux kernel source.

Soundcard radio modem driver

Syntax:

soundmodem=iobase,irq,dma[,dma2[,serio[,pario]]],0,mode

All parameters except the last are integers; the dummy 0 is required because of a bug in the setup code. The mode parameter is a string with syntax hw:modem, where hw is one of sbc, wss, or wssfdx, and modem is one of afsk1200 or fsk9600.

The line printer driver

'lp='

Syntax:

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

You can tell the printer driver what ports to use and what ports not to use. The latter comes in handy if you don't want the printer driver to claim all available parallel ports, so that other drivers (e.g., PLIP, PPA) can use them instead.

The format of the argument is multiple port names. For example, lp=none,parport0 would use the first parallel port for lp1, and disable lp0. To disable the printer driver entirely, one can use lp=0.

WDT500/501 driver

Syntax:

wdt=io,irq

Mouse drivers

'bmouse=irq'

The busmouse driver accepts only one parameter, that being the hardware IRQ value to be used.

'msmouse=irq'

And precisely the same is true for the msmouse driver.

ATARI mouse setup

Syntax:

atamouse=threshold[,y-threshold]

If only one argument is given, it is used for both x-threshold and y-threshold. Otherwise, the first argument is the x-threshold, and the second the y-threshold. These values must lie between 1 and 20 (inclusive); the default is 2.

Video hardware

'no-scroll'

This option tells the console driver not to use hardware scroll (where a scroll is effected by moving the screen origin in video memory, instead of moving the data). It is required by certain Braille machines.

SEE ALSO

klogd(8), mount(8)

Large parts of this man page have been derived from the Boot Parameter HOWTO (version 1.0.1) written by Paul Gortmaker. More information may be found in this (or a more recent) HOWTO. An up-to-date source of information is the kernel source file *Documentation/kernel-parameters.txt*.

COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man-

pages/.