## NAME

monitor, edparams − load and start Minix, modify boot parameters

## SYNOPSIS

**/boot**
**edparams** *device* [*command* ...]
**boot.com** *virdisk* [*command* ...]

## DESCRIPTION

This text describes the Boot Monitor, a boot time interactive program designed not only to load and start Minix, its most important task, but to also provide an interface to configure Minix and to boot other operating systems.

The monitor is controlled with an environment that is modeled after the Bourne shell. This environment is filled at startup with default values that depend on the machine the monitor is running on and the environment settings saved into the boot parameters sector (the second sector on a device). When the environment is loaded, the monitor executes the function named **main**, which by default starts a simple menu.

The environment can be manipulated at boot time from the monitor prompt, but may also be edited using **edparams** on a given device. **Edparams** simulates the monitor as much as it can, echoing commands it can't execute between brackets. It can also be used in Makefiles and scripts by giving it commands as arguments.

The DOS version of the monitor, usually named **boot.com** under DOS, boots Minix from a "DOS virtual disk". **Boot.com** is a simple COM program that interprets a DOS file as a disk, loads a Minix kernel from the active partition in the same way as the BIOS based monitor, and executes it to start Minix. All the monitor commands function in the same way, except for the **boot** command, it can only load Minix. The monitor grabs as much free memory as it can for Minix to work in, as the **memory** variable shows. Further details on how to run Minix under DOS, Windows 95, or even Windows NT are written down in **dosminix**(8).

## COMMANDS

The monitor is best described by the commands you can type to the '>' prompt. This is known as the "monitor mode". You can enter this mode by hitting the Escape key. These are the monitor commands:

*name* = [**device**] *value*

Set environment variable.
Changes the value of *name* to *value*. The optional word **device** marks *name* as being subject to device translation. (See the section on devices.) These (name, value) pairs are passed to the kernel who uses them to configure itself. These variables are passed by default:

**rootdev**
This is the device used as your root device. It is by default set to **ram,** which means that the device specified by **ramimagedev** will be loaded into the RAM disk and used as root. If you change this variable then a physical device will be used as root, and the RAM disk will be uninitialized and have the size specified by **ramsize**.

**ramimagedev**
Describes the device to use to initialize the RAM disk if **rootdev** is set to **ram**. It's by default set to **bootdev**, a special name for the device the monitor booted from.

**ramsize**
The size of the RAM disk. If the RAM disk is used for the root file system then the root file system is stretched out to **ramsize** if possible.

**processor**
Set by default to **86**, **186**, **286**, **386**, **486**, ... depending on the hardware you have. You can set it to a smaller value to test your kernel in a more limited environment.

**bus**

The type of system bus, either **xt**, **at** or **mca**. This answers basic questions like: "How many interrupt controllers and how to initialize?" Or: "Does the keyboard have LEDs?"

**memory**
List of memory free for use by Minix. It is a comma separated list of *base:size* pairs denoting the byte offsets and sizes of free memory in hexadecimal. **800:925E0,100000:F00000** is a typical example of about 585K starting at 2K, and 15M starting at 1M. (The first 2K are BIOS parameters and the 53K under the 640K boundary is the monitor itself.) The very last number you can play with if you know what you are doing. Either increase it if the monitor has it wrong, or decrease it to test if Minix still runs with less memory then normal.

**video**
Describes capabilities of the VDU: **mda**, **cga**, **ega** or **vga**.

**chrome**
Either **color** or **mono**.

**c0**
By default **at** (AT compatibles), **bios** (XT or PS/2), or **dosfile** (running under DOS). The **c0** variable binds a driver to the first controller, i.e. the **/dev/c0*** devices. The monitor sets **c0** to a suitable default, so that most machines can find their disk.

**console**
If set to a hexadecimal value it makes the monitor set the BIOS video mode to this value when Minix is started. This allows the use of video modes with more rows or colums than the standard 80x25 mode. You can use any text mode in the 00-FF range, and VESA extended modes in the 100-FFF range. Most text modes use a 9x16 font with 400 scanlines on screen, so you see 400/16 = 25 lines. The text mode can be modified by adding special flags to the console setting. Add 2000 to switch to 480 scan lines, adding 20% more lines to the screen. Add 4000 to select a 9x14 font, so 28 or 34 lines are shown. Add 8000 instead to select an 8x8 font showing 50 or 60 lines. Each setting has drawbacks. Using 480 scanlines implies a 60 Hz refresh, so the screen may flicker. The 8x8 font looks squashed. More letters on screen require more memory, so there is less for virtual consoles. Interesting modes to try are 4003 (80x28), 2003 (80x30), 6003 (80x34), 8003 (80x50), A003 (80x60), 109 (132x25), 10A (132x43), 10B (132x50), 10C (132x60). The 109 VESA mode is often available, and can be modified like mode 3. Use mode 7 instead of 3 for monochrome. Which modes and flags work can only be found out by experiment. More parameters may follow the mode number that are of interest to the console driver, see **boot**(8).

**dosfile-d0**
Set by the DOS version of the monitor to the name of the virtual disk, i.e. the *virdisk* argument as shown above. The "dosfile" driver will use this as the name of the file to use as a disk.

Two variables are only used by the monitor, even though they are passed to the kernel too:

**image**
The name of the file containing the kernel image, by default **minix**. If it refers to a directory however then the newest file inside the directory is chosen to be the kernel image. The names inside **/minix/** are best set to the Minix version you are using, which looks good when the monitor prints its name. Rules for pretty printing image names:

A '/' or '_' is changed to a space.

The first letter is changed from lowercase to uppercase.

An 'r' if followed by a digit changes to " revision ".

**label**

If set then only processes marked with this label or without a label are loaded from the image.

**Installboot −boot** will create functions to select images and labels. These functions will set **label** and **image** and echo what you selected. The two numbers separated by a colon used as an image name tell the starting sector and sector count of the image on disk.

*name*() *command*

Define function.
Functions may be used to bundle a set of commands, so that you can easily boot Minix with a different set of parameters then normal. E.g.

ram() { rootdev=ram; boot }

will allow you to run Minix with the root device on RAM for a change, if you normally use a real device as root. There are three predefined functions, **leader**, with default value an **echo** command that shows the monitor's startup banner, **main**, with default value **menu**, and **trailer**, with default value a command that clears the screen. The monitor executes **leader;main** at startup to show the banner message and a menu. The **trailer** function is executed just before Minix is started. These three functions can be redefined as you please.

*name*(*key*) *command*

Define kernel selecting function.
The menu command uses functions like these to add menu entries to select a different kernel from a boot disk. **Installboot −boot** produces these functions when the images are labeled. The label **AT** would give:

AT(a) {label=AT;image=42:626;echo AT kernel selected;menu}

With the menu option:

a Select AT kernel

Typing **a** will then execute the **AT** function above.

*name*(*key*,*text*) *command*

User defined menu option.
This variant may be used to make any menu entry you like:

dos(d,Boot MS-DOS) boot d0p0

*Text* may be anything, even parentheses if they match.

*name*

Call function.
If *name* is a user defined function then its value is expanded and executed in place of *name*. Try a recursive one like 'rec() {rec;xx}' one day. You can see the monitor run out of space with nice messages about using **chmem**(1) to increase it's heap.

**boot** [−*opts*]
**boot** *device*

Boot Minix or another O.S.
Without an argument, **boot** will load and execute the Minix image named by the **image** variable. With options the variable **bootopts** is first set to −*opts* before Minix is started, and unset when Minix returns. With a *device* argument, **boot** loads the boot sector of *device* into memory and jumps to it, starting another operating system. You would normally use partitions on the first hard disk for this command (d0p[0−3]), using d0 will also work (choosing the active partition). One can also boot devices on the second hard disk (d1, d1p[0−3]) if the bootstrap writer did not hardwire the disk number to disk 0.
Some Operating Systems can only be booted from the active partition, if you use a '*', e.g. **boot *d0p2**, then partition 2 is first made active. You'll then need to use

**installboot −m /dev/c0d0 /usr/mdec/jumpboot** *keys*

with *keys* chosen so that Minix is booted at startup. (See **installboot**(8).)

**ctty** *n*

Copies output to and takes input from serial line *n* (0-3) at 9600 baud, 8 bits, no parity. This allows you to control a Minix system remotely through an RS-232 connection.

**delay** [*msec*]

Delay (500 msec default).
Fast booting speed was one of the objectives when this program was created, so a hard disk boot usually takes only a fraction of a second. If you need some time (to hit Escape, or stare at the numbers) you can use **delay** to make the monitor pause for a specified number of milliseconds.

**echo** *word* ...

Print these words.
Used to display messages, like the startup banner. Echo normally prints the words with spaces in between and a newline at the end. Echo understands special '\' escape sequences as follows:

| | |
|---|---|
| \ | (At the end) Don't print a newline. |
| \n | Print a newline. |
| \v | Print the monitor's version numbers. |
| \c | Clear the screen. |
| \w | Wait until a RETURN is typed |
| \\ | Print a backslash. |

**ls** [*directory*]

List contents of a directory.
Useful when looking for kernel images.

**menu**

Menu driven startup.
This command allows you to execute functions defined with a *key*. If no menu functions have been defined then **menu** will use this one hidden built-in function:

```
*(=,Start Minix) boot
```

Kernel selecting functions only add new options to this set, but if you define a two argument function yourself then the above one is no longer shown, allowing you to customize the menu completely. Your first function definition should therefore be one that starts Minix.

Menu entries are shown in the same order as **set** shows them. If you don't like the order then you have to unset the functions and retype them in the proper order.

If you type a key then a scheduled trap is killed and the appropriate menu function is executed. If you need more time to choose then hit the spacebar. A key not on the menu also kills a trap, but does nothing more.

**save**

Save environment.
This will save all the environment variables and functions with nondefault values to the parameter sector (the second sector on the boot device), so they are automatically set the next time you boot the monitor.

**set**

Show environment.
Show the current values of the environment variables and functions. Default values are shown between parentheses to distinguish them from values that were explicitly set.

**trap** *msec function*

Schedule function.
Schedules a function to be executed after *msec* milliseconds. Only the monitor mode cannot be interrupted, a scheduled trap is killed when the prompt is printed. Example:

```
main() {trap 10000 boot; menu}
```

This gives you 10 seconds to choose a menu option before Minix is booted.

**unset** *name* ...
Unset environment variables.
Removes the named variables and functions from the environment, and sets special variables back to their default values. This is also the only way to remove the "device name translation" property from a variable.

**exit**
Exit the monitor.
Reboot the machine, exit to Minix or exit to DOS as appropriate.

**off**
Turn the PC off.
If the PC supports power management then turn it off, otherwise print some error messages and do nothing.

**{** *command*; ... **}**
Bundle commands.
Treat a number of commands as a single command. Used for function definitions when a function body must contain more than one command.

## DEVICES

The Minix kernel can't do anything with device names, so they have to be translated to device numbers before they are passed to the kernel. This number is found under the st_rdev field (see **stat**(2)) of the file on the boot file system. The monitor will look for the device file with the working directory set to '/dev'. If it can't find the device name then it will translate names like 'ram', 'fd1', 'c0d1p0', 'c1d0p2s0', and even the obsolete 'hd2a' to what it itself thinks the numbers should be.

The special name **bootdev** is translated to the name of the device booted from, like 'fd0', or 'c0d0p1s0', and then searched for in /dev. **Bootdev** can only be translated to a device for the first controller, and only if the disks on that controller are numbered without "gaps". (The master device on the second IDE channel is always d2 on Minix. The BIOS will call it disk 0, 1, or 2 depending on the number of disks on the first IDE channel.)

Controller numbers are meaningless to the BIOS, so everything is assumed to be attached to controller 0. You can omit **c0** for device names, and it is best to always omit **c0** for the **boot** command, and to always use the full name for variables passed to Minix.

## EXTENSIONS

A few extensions have been made to this program for kernel hackers. They may be triggered by setting bits in the flags word in the kernel startup code (the mpx file.) The flag bits are:

0x0001 Call kernel in 386 mode.

0x0002 Do not make space for the bss areas of processes other than the kernel.

0x0004 Use the stack size set by **chmem**(1).

0x0008 Load MM, FS, etc. into extended memory.

0x0010 No need to patch process sizes into the kernel.

0x0020 The kernel can return to the monitor on halt or reboot.

0x0040 Offer generic BIOS support instead of just INT 13 (disk I/O).

0x0080 Pass memory lists for free and used memory (processes).

0x0100 Kernel returns monitor code on shutdown in boot parameters array.

## SEE ALSO

**controller**(4), **installboot**(8), **usage**(8), **boot**(8), **dosminix**(8).

## BUGS

The **delay** command will hang forever on the original IBM PC (not the XT!). Not that it matters, as everything takes forever on that box.

By redefining **leader** one can easily hide the identity of this program.

## ACKNOWLEDGMENTS

Earl Chew, for the inspiration his ShoeLace package provided, unless he wants to file a "look and feel" suit against me, then I will say I modeled it after the Sun ROM boot monitor, which is also true.

## AUTHOR

Kees J. Bot (kjb@cs.vu.nl)