NAME

usage − installing and using MINIX

DESCRIPTION

This manual page describes the installation and use of MINIX from a System Administrators point of view. It contains an installation guide, instructions on how to do the initial configuration and some other info. Please read this document entirely before attempting to install MINIX. The installation steps are in the proper order, but not all the information you may need is presented at the right moment. Other detailed information that may be useful can be found in **boot**(8), **hier**(7), and in **dosminix**(8) if you run MINIX under DOS.

1. MINIX UNDER DOS

Installation of MINIX to run under DOS is a nonevent. Chances are, you are reading this manual page from an already running MINIX system, but if not then the setup goes like this:

Unpack the DOSMINIX.ZIP file using one of the popular ZIP utilities, such as PKZIP or WinZIP. Next reboot Windows and hit F8 just when you see the "Booting Windows" message. From the menu that appears choose "Command prompt only", or if that doesn't work "Safe mode command prompt only". Under Windows Me you can use a startup disk to boot the PC into DOS. Move to the directory containing the MINIX files and type:

boot minix.mnx

Type '=' and presto, you are running MINIX. Most of the rest of this manual, which deals mainly with running MINIX in a true hard disk partition, does not apply to you. Your system is already installed completely, with all binaries and sources present, so all the talk about getting MINIX on your disk can be skimmed over. Pay attention again when the focus shifts to the configuration of the system. Section 9 is where this happens first. (The main challange to a DOS installation is to figure out which parts of the installation manual do not apply.)

2. REQUIREMENTS

The minimum system MINIX can be installed on comfortably is an IBM PC/AT or PS/2 with a 286 processor, 2 MB memory, a 720 kb diskette drive, and 35 MB free space on an AT, ESDI, or SCSI hard disk (the latter controlled by an Adaptec 1540.) MINIX for the 386 (MINIX-386 for short) can be installed on a machine with at least a 386sx processor, 3 MB memory and at least 35 MB of disk space.

The minimum system MINIX can be installed on **un**comfortably is an IBM PC/XT with 640 kb memory. MINIX-386 can more or less run in 2 MB memory. See sections 16 and 17 on "low memory" issues.

3. MINIX INSTALLATION BACKGROUND

The objective of the installation is to create a partition on your disk and to put MINIX into it. MINIX really requires at least two partitions however, so the single "primary" partition is split into two or three subpartitions. The **s0** subpartition will contain the root file system, the **s1** subpartition may optionally contain swapspace, and the **s2** subpartition will contain the **/usr** file system. What Windows calls "drives", i.e C:, D:, E:, MINIX calls "file systems". MINIX does not use drive letters, but requires that one file system is made a part of another file system by "mounting" one on the other. The "root" file system is always present and starts with the directory "/", the root of the directory tree. The root file system contains a few programs in **/bin**, device files in **/dev**, and configuration files in **/etc**. This is just enough to get the system started. MINIX will soon extend its directory tree by mounting a file system on the **/usr** directory. What is henceforth known as the /usr file system contains all MINIX programs in **/usr/bin**, file system sources in **/usr/src**, etc, etc. The ROOT.MNX image contains the complete MINIX root file system, but USR contains just a small subset of the /usr file system, with just enough utilities to install MINIX. The complete /usr file system is split up into the USR.TAZ, SYS.TAZ and CMD.TAZ archives that are installed later to fill /usr.

Let's suppose your first hard disk, which has device name **/dev/c0d0**, has Windows already present in the first primary partition (**/dev/c0d0p0**), and some free space left after that. After MINIX is installed in that free space the disk will look like this:

/dev/c0d0	Whole hard disk #0
/dev/c0d0p0	Windows C: drive
/dev/c0d0p1	MINIX primary partition
/dev/c0d0p1s0	MINIX root partition
/dev/c0d0p1s1	MINIX swap partition (optional)
/dev/c0d0p1s2	MINIX /usr partition

/dev/c0d0 is the sum of a partition table, /dev/c0d0p0 and /dev/c0d0p1. Likewise is /dev/c0d0p1 the sum of a subpartition table, /dev/c0d0p1s0 and /dev/c0d0p1s2. Read the "DEVICES" sections for more information on MINIX devices.

4. INSTALLATION

You can install MINIX automatically or manually as described in the sections below. The end result is the same, but manual installation allows one to deviate from the preconfigured choices. You may wish to read the manual pages of the programs used below before you start. You may especially want to read **boot**(8) if your machine is different from what the majority buys, because you may need to set a few boot parameters to configure drivers. To do this type **ESC** to get to the Boot Monitor prompt, set the appropriate variables, use **save** to store the settings and **menu** to continue where you left off.

To install the system you need two diskettes: a bootable root diskette and a diskette full of binaries to use as **/usr**. These diskettes are named **ROOT** and **USR**. These two diskettes may also be combined on a single high density diskette. In that case the USR part is on the **p2** partition.

Insert the ROOT diskette, boot the machine and type '=' to the menu. The MINIX kernel is loaded and takes control when you see the copyright banner. After loading the root diskette into the RAM disk you will be asked to finish the name of the device to mount on **/usr**. Type **fd0p2** for a diskette that contains both ROOT and USR, otherwise replace ROOT by USR and type **fd0**. Login as **root**.

5. AUTOMATIC INSTALLATION

Before starting the installation, you must either have a free partition available or have at least 35 MB not in any partition so you can create a MINIX partition.

Type **setup** to start the installation script. First it offers to install a national keyboard map. The names should be clear, except for **us-swap**, which swaps the CTRL and CAPS LOCK keys of a standard US style keyboard for people who believe that the natural place of CTRL is next to A. The default suggested between [and] is the US standard keyboard.

The next thing to do is to make a partition, for this you are placed in a partition table editor named **part**. This partition table editor is very easy to use (in the author's opinion), but you will probably hate it. You can move all over the place with the arrow keys, change values, and make a mess of your partition table real quick. So if you get into trouble, type 'q' to quit, 'n' to not write the table, and RETURN to start over. Use the '?' key to get help.

With the '+' and '−' keys you can select the disk device to install on, probably **/dev/c0d0**, the first hard disk. Type 'r' to load the partition table of the selected disk. Either create one new partition by modifying a partition marked "None", or reuse an existing partition by changing its type to "MINIX" (hex code 81). DO NOT use part to shrink an existing partition! It will destroy all data in that partition. MINIX needs a partition of at least 20 MB, but not larger than 128 MB (MINIX-86) or 1 GB (MINIX-386). The system needs 35 MB in compiled state.

The script then wants to know the name of the partition you've created. The partition name is probably still visible on the screen. Combined with the drive name you have to type c0d0p1, c0d2p0 or something.

The next question is the amount of swapspace you want to give MINIX. There will be a suggested value based on the amount of memory your system has. If you have more then enough memory then don't bother with swap. MINIX doesn't handle it very well yet, or ever, only memory starved systems need it so that **make world** can run.

The new partition table is reloaded into the disk driver, and the new MINIX partition is carved up into two or three subpartitions, a 1440 kb root, maybe some amount of swap, and the rest for /usr.

After making /usr, it is immediately put to use to replace the installation /usr file system so that you can remove the USR diskette and insert the ROOT diskette (unless they are one and the same). The root file system is filled with the contents of the ROOT diskette and slightly patched up to work on the hard disk (/etc/fstab.)

You can now skip the next section and move to "TESTING", but it may be instructive to read it anyway.

6. MANUAL INSTALLATION

The instructions that follow are at a very low level and require you to be very careful. The big advantage is that you know precisely what tools have been used and how everything works. The disadvantage is that you may easily make a mistake that either forces you to start over if you are lucky, or wipes out the contents of your hard disk if you are not. Only if you really want to do something different should you use a manual installation. Slavishly following the steps shown below will only make you end up with the same result as an automatic installation.

Run **part** to make partitions to load the system into. The best thing to do is to make one large primary partition of type "MINIX" and to carve this partition up into three subpartitions for root, swap and /usr. The assumption is that you will use the second partition on the first hard disk, **/dev/c0d0p1**, and that **c0d0p1s0** is the root subpartition, **c0d0p1s1** is swap and **c0d0p1s2** is /usr. If you want to use the first partition on the second hard disk for instance, then substitute c0d1p0 and c0d1p0s[012] for the above. See the section on devices below, and the manual pages of **part**(8) and **controller**(4). Start **part** and select the disk that you want to install MINIX onto. In our example it will be **/dev/c0d0**.

Use **part** to make a single partition in the primary partition table of type "MINIX", then hit '>' on this new partition to make a subpartition table.

For the root subpartition you are advised to use 1440 kb exactly. You can make it larger if you want to, but it is advisable never to let the contents outgrow a floppy. (The ROOT diskette is a copy of a root file system, and will be used to fill your root subpartition.)

The second subpartition is for swapspace. You can use it to enlarge the amount of total memory (real + swap) if your system has less than 2M (16-bit mode) or 4M (32-bit mode). Note that only one MINIX swap partition is needed on your system, so if you have another MINIX partition then you can use its swap partition instead.

Use the rest of the partition for **s2**, the /usr subpartition.

When you are done check that /dev/c0d0p1s0 is active (the * after the partition number) so you can boot from it later.

If your disk has bad blocks then don't put the root or swap subpartition on top of them. Make sure the inode tables in the other partitions don't have bad blocks either. You can put the subpartitions out of order on the disk if that helps. Subpartition tables, unlike the main partition table, are not sorted by the driver.

After making the partitions you do not have to reboot. The disk driver reloads the partition tables on the next access if the disk is not in use. (Open or mounted.)

To be able to boot from /dev/c0d0p1s0 you must place a master bootstrap in /dev/c0d0p1. It has been placed there by **part** if it told you that it was creating a new partition table, but

```
installboot -m /dev/c0d0p1 /usr/mdec/masterboot
```

will put it there for sure.

Let's start by initializing the swap partition first, if you allocated one. We may need it already, so mount it.

```
mkswap /dev/c0d0p1s1
mount -s /dev/c0d0p1s1
```

Next make a file system for on-disk /usr and copy the floppy /usr on to it.

mkfs /dev/c0d0p1s2
readall −b /dev/c0d0p1s2 | sh
mount /dev/c0d0p1s2 /mnt
cpdir −v /usr /mnt

This will create a file system on /dev/c0d0p1s2, mount it on /mnt, and copy the contents of the USR floppy onto it. The call to **readall** marks bad blocks on the file system as unusable, you can omit this on a drive known to be spotless (IDE or SCSI.)

You can now use the new /usr in place of the USR floppy:

umount /dev/c0d0p1s2
umount /dev/fd0 # fd0p2 if combined
mount /dev/c0d0p1s2 /usr

This little dance has freed up your floppy drive, so please remove the USR diskette and replace it by the ROOT diskette. Make a file system for the root with at least 512 inodes (files), and fill it from the floppy:

mkfs −i 512 /dev/c0d0p1s0
mount /dev/fd0 /fd0
mount /dev/c0d0p1s0 /mnt
cpdir −v /fd0 /mnt
umount /dev/fd0

Remove **/mnt/etc/issue** to get rid of the "use setup" message that greets you when you boot, and edit the file **/mnt/etc/fstab** to name the devices MINIX has been installed on. In our example it should look like this:

root=/dev/c0d0p1s0
swap=/dev/c0d0p1s1
usr=/dev/c0d0p1s2

Unmount the new root:

umount /dev/c0d0p1s0

Make it bootable:

installboot −d /dev/c0d0p1s0 /usr/mdec/bootblock boot

The automatic script would now set the **rootdev** and **ramimagedev** boot variables. You can do this now using the **edparams** command, but it is easier to postpone it until the testing phase. The settings should be:

rootdev=c0d0p1s0
ramimagedev=c0d0p1s0

7. TESTING

By now a new MINIX system is present on your hard disk. Time to see if it works. Leave the ROOT diskette in the drive and type **halt**. You are now going to use the power of the Boot Monitor on the diskette to boot the MINIX partition on the hard disk. Use the monitor command **boot c0d0p1** to boot the primary partition MINIX has been installed in. (It is "c0d0p1" in our example.)

The hard disk bootstrap is now showing the menu again. You can type '=' to start MINIX, but you probably want to change the boot parameters. Hit **ESC** once more to get to the command prompt. The command **set** shows what the current parameters are. Here is an example that shows how to make a menu to either start MINIX or boot Windows:

minix(=,Minix) boot
win(w,Windows) boot c0d0p0
save

Windows is assumed to be in the first partition in the example above (c0d0p0). When finished type **menu** to see if the menu looks right. If so hit '=' to start MINIX. Log in as root.

8. ADDING PROGRAMS AND SOURCES TO /usr

The **setup** command can also be used to add files from floppy sets to the system. The **USR.TAZ** (programs and stuff), **SYS.TAZ** (system sources), and **CMD.TAZ** (commands sources) are all installed relative to the **/usr** directory, so the command to use three times is

setup /usr

Setup will ask for the size of data on the floppies, which is by default simply the entire floppy. You will see some "Cannot make directory" errors while extracting, as some directories already exist. Ignore these messages. You need the **USR.TAZ** set if you want a working MINIX system, **SYS.TAZ** if you want recompile the system or study it, and **CMD.TAZ** if you also want the sources of the commands. On a disk space starved machine you could opt to do without the commands sources, as they are not absolutely necessary to understand MINIX.

If your machine does not have enough memory to run **setup /usr** then type these commands manually:

cd /usr
vol /dev/fd0 | zcat | tar xvfp −

If **USR.TAZ** is already present on the hard disk in an DOS or Windows partition, then this command can be used under MINIX-386 to extract it to avoid the floppy detour:

cd /usr
mtools copy c0d0p0:USR.TAZ − | setup /usr

In 16-bit mode you don't have mtools, but maybe dosread will work:

cd /usr
dosread c0d0p0 USR.TAZ | setup /usr

The file doesn't have to be in the root directory of **c0d0p0**, of course, **c0d1p0:/TMP/USR.TAZ** would name a file on the first partition of the second hard disk in the directory **\TMP**.

The /usr file system can also be filled through a network from a remote host if MINIX if you can get networking going with the NET.TAZ supplement. Follow the instructions in **boot**(8) to configure TCP/IP and boot MINIX. There are now two ways to fill /usr. One is to add the host name and login name of a remote host and a remote user to **/usr/src/.rhosts**, as root, and to use the following command on the remote host:

rsh −l root *minix-box* **setup /usr < USR.TAZ**

Two is to use **urlget** to copy the data directly from a Web or FTP site by using these commands under MINIX:

cd /usr
urlget *url*.../USR.TAZ | setup /usr**

The sources may be installed using exactly the same commands, but with **USR.TAZ** replaced by **SRC.TAZ**. Note that this means that the sources must also be extracted relative to **/usr**.

9. NAMES

A standalone machine will have to be given a name. As **root** type

echo *name* **>/etc/hostname.file**

to change the host name of your machine to *name*.

10. ACTIVE ON BOOT

You may want to make the MINIX partition active so that it is automatically booted. With Windows **fdisk** or MINIX **part**, mark the primary partition that contains MINIX active. Using the menu you made earlier you can boot either MINIX or Windows at a keypress. You can even set timeouts. To boot MINIX automatically after 5 seconds:

main() {trap 5000 minix; menu}

See **monitor**(8) for all the details on the monitor.

If you don't trust this then you can rig up a diskette that boots the MINIX partition when left in the drive:

installboot −m /dev/fd0 /usr/mdec/jumpboot 010

The numbers 010 indicate the device (disk or partition) that must be booted, i.e. **/dev/c0d0p1s0** in this example. Take the name of the device, and use the disk, partition and subpartition numbers, or less. So c0d1p2s0 -> 120, c0d3 -> 3, c0d2p0 -> 20.)

11. DEVICES

A crash course on the MINIX devices in **/dev**: The first two hard disks are named **c0d0** and **c0d1**. These devices address the entire hard disk, from the first to the last byte. Each disk has four partitions, for disk 0 they are **c0d0p0**, **c0d0p1**, **c0d0p2**, and **c0d0p3**. And for disk 1 they are named **c0d1p0** to **c0d1p3**. These partitions may contain file systems, **c0d0p0** often contains the MS-DOS or Windows "C:" file system. MINIX can use these partitions for file systems too, but you can also partition one of these "primary partitions" into four so-called "subpartitions". The subpartitions of **c0d0p0** are named **c0d0p0s0**, **c0d0p0s1**, **c0d0p0s2**, and **c0d0p0s3**. The other partitions may have four subpartitions that are named in the same way. See **controller**(4) for an elaborate description.

You may need to add devices to **/dev**, because not all devices are present to keep down the clutter. The command **MAKEDEV** knows how to make devices, and **DESCRIBE** can tell you what an unknown device may be, or even what all devices in **/dev** may be if called without arguments. Devices are described in **dev**(4), with pointers to more specific pages.

12. EDITORS

The editors available are **elvis** (a **vi** clone), **elle** (a simple **emacs** clone), and the old MINIX **mined** editor. Of these editors only elvis can recover your file after a system crash. Only **mined** is available at installation time. (All you need to know about mined right now is that CTRL-X gets you out of it.)

13. BOOT MONITOR VS. MINIX

The Boot Monitor uses the BIOS to address disks, so it has no idea of controllers, it just lumps everything together and ignores controller numbers. So what the monitor thinks are **d0**, **d1**, and **d2**, may be **c0d0** (IDE primary master), **c0d2** (IDE secondary master), and **c1d3** (SCSI disk at target 3). One must keep this in mind when MINIX is installed on a disk other than the very first. So if MINIX is installed in the third partition of the SCSI disk, then **boot d2p2** will boot it, and **rootdev=c1d3p2s0** will tell MINIX where its root file system is.

14. NATIONAL KEYBOARDS

The directory **/usr/lib/keymaps** contains keymap tables for several national keyboards. If you have a German keyboard for instance, then

loadkeys /usr/lib/keymaps/german.map

will load the German key translation table into the keyboard driver. Copy the map to **/etc/keymap** once MINIX is installed on the hard disk, because having to type a key sequence like one of these:

loadkezs −usr−lib−kezmaps−german.map
loqdkeys =usr=lib=key,qps=french.,qp

on a reboot gets a bit annoying after a while. Send corrections and new keymaps to the person named below. (Do not send a Dutch keymap, buy yourself a real keyboard instead.)

SUGGESTIONS

Below are a few useful suggestions. Some of the information can be of use in other situations than described here.

15. VIRTUAL CONSOLES

Hold down the ALT key and press the left or right arrow key, F1, or F2. This switches the console between two login sessions. (Unless you have an old mono adapter, because virtual consoles sit in video memory, and a mono adapter only has memory for one.)

Note that kernel messages, including function key output, only appear on the first console. This may be confusing, but it keeps the other consoles clean.

16. LOW ON MEMORY

The normal installation requires that you have enough memory for a large RAM disk. You can still install MINIX normally if you either have a high density diskette drive for a combined root+usr floppy, or you have two floppy drives of at least 720 kb. Before booting you have to set the variable **rootdev** to the same value as **ramimagedev**. This is slower then a RAM disk, but saves a lot of memory.

The automatic installation script knows how to handle this new situation. If you install manually then you have to use

cpdir −vx / /mnt

to copy the root device to disk. When it is time to fill /usr and you only have one floppy drive then hit DEL to get out of the installation script and reboot as described in "TESTING". You can then finish the installation manually.

17. LOW ON MEMORY AND ONLY ONE 720 KB FLOPPY DRIVE

If you only have one 720 kb floppy drive and your system is low on memory then you can use the TINYROOT.MNX boot image. This image contains a small kernel with only the BIOS disk driver, and a small root file system. You can use this disk to boot your machine. Use the normal ROOT.MNX to install the root file system. Keep booting your machine with TINYROOT until you have compiled a small kernel for your system. Use the **rootdev** boot variable to select the hard disk root file system. Do **not** use TINYROOT for anything other than booting, always use ROOT when mentioned.

18. FLOPPY DRIVE 1 IS A HIGH DENSITY DRIVE

If you would like to install from floppy drive 1 then you need to copy at least one sector from the USR image onto a diskette for drive 0. The USR bootstrap has been rigged to boot the other drive.

19. INSTALLING ON A SECOND HARD DISK

MINIX doesn't care if it is installed on the second disk of a system with two disks. The only problem is to get it booted. You can either rig up a diskette to boot MINIX as shown earlier, or you can use the same trick on the first disk. The command

installboot −m /dev/c0d0 /usr/mdec/jumpboot 1

will lock the first disk into booting the second disk. Note that this command modifies the disk outside a MINIX partition, overwriting a bit of code that has likely been put there by Windows fdisk. First verify that the Boot Monitor can boot a Windows partition, because then the MINIX master bootstrap can do it too.

20. LOTS OF MEMORY ON A 286

You will have a hard time making MINIX use up 3 MB memory. Memory you can spare can be used for a "second level block cache" on the RAM disk. The File System uses the second level cache to store copies of disk blocks that are pushed out of the normal (primary) block cache. The size of the primary cache is compiled into the FS server, but the size of the second level cache can be set with the **ramsize** boot variable. Set it to a number between 0 and 512. 512 kilobytes is enough to keep most of the compiler cached.

21. LOTS OF MEMORY ON A 386+

Processes can be as big as you would like on a 386, but in practice 4 MB is a lot, and 8 MB is infinite. The installation script sets up a second level cache for MINIX-386 of up to 1024 kilobytes. This is because the default file system cache is only 80 kb. Your first point of call is to get rid of the poorly performing second level cache by setting **ENABLE_CACHE2** to 0 and to assign the memory used by it to the normal block cache by enlarging the appropriate **NR_BUFS** and **NR_BUF_HASH** constants in <minix/config.h> with as much as you can spare. (1024 for NR_BUFS is the minimum to keep **cc −c** cached. 2048 is then a nice value for NR_BUF_HASH.) Disable the second level cache, compile a new kernel, reboot and set **ramsize** to 0.

22. LOTS OF DISK SPACE

The maximum file system size is 1 GB for MINIX-386 and 128 MB for MINIX-86. (MINIX-86 can handle larger file systems, but **fsck** can't check them.) Note that a MINIX file system can only contain 65535 inodes (files), so the average file should be 16 kb to completely fill it. It may be better to make two smaller file systems. Besides, fsck takes forever on a large file system.

SYSTEM ADMINISTRATION

The system has been set up with the idea that working as root is a bad thing to do. As root you are in no way protected from doing stupid things. So don't do development as root, but work as **bin**! Only in exceptional cases do you want to become root. Being root is fun for wannabe hackers; administrators know better.

To make life easier for bin, some programs like **su**(1), **install**(1) and **shutdown**(8) treat bin and other members of the operator group as special and allow them the privileges of root. (One is an operator if one's group id is zero.) Operators should share the shadow password of root by having **##root** in their password field. This way they all have one face (password) to the outside world, forming no greater security risk than root alone.

The home directory of bin contains one important Makefile. You can use it to recompile all the commands and libraries of the system. Type **make** to see the usage message. If you want to compile just one command then you can simply type **make** to do so. To put it in its proper place you have to type **make install**. Read the Makefiles in the **commands** and **lib** subdirectories to understand how everything is put together. If you are tight on memory then **make** may fail to traverse down the source tree and also compile things. You will have to type **make** in each subdirectory. You can run make in /usr/src at the end to see if you've missed something or not.

The shell used by MINIX is a minimal version of **ash**, the BSD shell. It has been modified to offer simple line editing using the **editline**(3) library.

The kernel is not compiled from the master Makefile. To make a new kernel you have to step into the **tools** directory. There you can run four different make commands:

make This makes all the different kernel parts and combines them in the file named **image**.

make fdboot
: As above and then makes a boot floppy that you can use to restart your system with. You are prompted for the floppy device name.

make hdboot
: First makes the image file and then copies it into the directory **/minix**. If there are already two images in that directory then the newest image will be removed to make space for this newer image. It is assumed that the oldest image is the most stable system image, one that always works, and that the newest image is experimental. Check beforehand what **/minix** contains before you run **make hdboot**. Remove the oldest image if you want another image to become the stable image. The Boot Monitor chooses the newest image in **/minix** to boot. You can use the monitor command **ls minix** to view the images present, and set the **image** variable to the full name of the image you want to use instead if the newest doesn't work. The images in **/minix** are named using the MINIX release and version numbers with an extra revision number added to distinguish the images.

The first new kernel you would like to make is one configured for your system. The kernel you are running now contains several drivers you don't need, or may be missing drivers that you might want. In <minix/config.h> you can find a number of **ENABLE_***XXX* variables that can be set to **0** to exclude, or **1** to include a particular driver. The full list of configurable parameters and what they do are described in **config**(8). It is invaluable in figuring out what to change and how in <minix/config.h>.

Configuring a new kernel is sometimes not enough to enable new devices, you sometimes need to use the **MAKEDEV** command to make new device files in **/dev**. For pseudo-ttys you also have to check if **/etc/ttytab** mentiones the new devices.

New additions to the system can be made in the **/usr/local** tree. An empty directory tree has been set up for you and binaries and manual pages are already in the search paths. You can make a new user entry with the **adduser** command.

The **TZ** variable in **/etc/profile** tells the time zone offset from the wall clock time to GMT. You have to change it for your time zone. (See **TZ**(5).)

The function keys produce debug dumps, showing various interesting data about the system. F1 lists processes and F5 shows ethernet stats, which may be of use now. Read **console**(4) to know all the details of the screen and keyboard.

23. SYSTEM SHUTDOWN

You can't just turn a MINIX system off. MINIX must be told to flush the modified data in the file system cache first. The following commands/keystrokes can be used to exit MINIX properly:

shutdown
First alert all users and then all processes of the impending shutdown then halt or reboot the system in one of various ways. See **shutdown**(8).

reboot / halt
Alert all processes of the system shutdown then reboot or halt.

CTRL−ALT−DEL
Halt the system by running **shutdown −h now**.

MINIX halts by returning to the Boot Monitor, MINIX reboots by instructing the monitor to reboot MINIX. (MINIX is just a subprocess to the monitor.) Either halt MINIX and use monitor commands to escape MINIX, or use **shutdown −R** to reset the system.

When exiting MINIX running under DOS the Boot Monitor's **exit** command will return you to the DOS prompt. The Boot Monitor and MINIX are together just a pretty big DOS program as far DOS is concerned.

FILES

/usr/ast Honorary home directory of Andrew S. Tanenbaum. Doubles as the place where the default setup for a new user is found.

SEE ALSO

dosminix(8), **monitor**(8), **boot**(8), **part**(8), **mkfs**(1), **mount**(8), **M**(8), **fstab**(5), **hier**(7), **config**(8), **console**(4), **dev**(4), **adduser**(8), **TZ**(5), **mkdist**(8), **shutdown**(8).
"Operating Systems − Design and Implementation 2/e" by Andrew S. Tanenbaum and Albert S. Woodhull.

NOTES

The notation **<*file*.h>** refers to a C language include file in /usr/include.

Root and **bin** do not have the current directory in their program search path to avoid executing programs left around by malicious people. This means that to run **foo** from the current directory, **./foo** must be typed.

BUGS

There are many PS/2 models, all different. Some will run MINIX, some won't, some crippled if you lie to MINIX by setting **processor** to **86**. Almost no PS/2 has a standard disk, so setting **c0** to **esdi** or **bios** will be necessary.

Except for the floppy driver, none of the DMA based drivers know about DMA being limited to a 24 bits address, i.e. the first 16 MB. So under MINIX-386 you run a slight risk that a **tar** or **dd** command may use a buffer above 16 MB for reading or writing to a character device. This only happens if the low 16 MB is taken by some huge processes, and you have more than 16 MB, of course.

AUTHOR

Kees J. Bot <kjb@cs.vu.nl>