

WHAT YOU NEED :

Boot Floppy (Root + Usr Floppy)

Set of floppies for USR.TAZ, SYS.TAZ, and CMD.TAZ.

Free Hard disk partition

INTRODUCTION :

Before installing Minix, you will have to transfer Minix on floppies. For any 386/486/586 system, you will first need to make a four sets of floppies of 1.44 MB.

Fdvol 1440 A: i386\ROOT i386\USR => This is a combination of ROOT + USR floppy

Fdvol 1440 A: i386\USR.TAZ => This is a base system (total 3 floppies)

Fdvol 1440 A: SYS.TAZ => System Resources (total 2 floppies)

Fdvol 1440 A: CMD.TAZ => Command Sources (Total 3 floppies)

To reconstruct a .TAZ file on a sequence of 1.44 MB floppies via drive A:, boot MINIX, log in, and type:

```
setup dir
```

Where 'dir' is the directory where a package is to be installed. Use 'install -d dir' to create the directory if it doesn't yet exist.

PROCEDURE FOR INSTALLATION:

The objective of the installation is to create a partition on your disk and to put MINIX into it. MINIX really requires two partitions however, so the single "primary" partition is split into two subpartitions. The a subpartition will contain the root file system, and the c subpartition will contain the /usr file system. What MS-DOS 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 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/hd0, has MS-DOS already present in the first primary partition (/dev/hd1), and some free space left after that. After MINIX is installed in that free space the disk will look like this:

```
/dev/hd0  Whole hard disk #0
/dev/hd1  MS-DOS C: drive
/dev/hd2  MINIX primary partition
  /dev/hd2a  MINIX root partition
  /dev/hd2c  MINIX /usr partition
```

/dev/hd0 is the sum of a partition table, /dev/hd1 and /dev/hd2. Likewise is /dev/hd2 the sum of a subpartition table, /dev/hd2a and /dev/hd2c. Read the "DEVICES" sections for more information on MINIX devices.

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.

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 fd0c for a diskette that contains both ROOT and USR, otherwise replace ROOT by USR and type fd0. Login as root.

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.

FIPS

FIPS is freely available from [Tucows](#) and the GNU archives. It is a dos based program that will allow you to resize your partitions on your harddrive. Make sure you read all the documentation that comes with FIPS. Mistakes made on your partition table can cause loss of data or even clear your whole harddrive.

If you are trying to split your existing partition from dos, win3.1x, or win95, you have to run scandisk (or chkdsk for old version dos) first. Then run defrag tool to defragment your current partition. For win3.1x and win95 users, you have to disable the virtual memory before defragmentation. You can find it from control panel, after you split your partition successfully then enable it back. There probably are some problems to split a winNT's partition or you want to make MINIX co-exist with your NT. I won't suggest you to do this. Since NT and MINIX always need to save their code of boot and partition table in the MBR (Master Boot Record). If you do want to install them on one machine, I suggest you to buy SysCommander or try Linux LiLo that can allow multiple OS on one system.

Notice for large size disk users (> 512 Mbytes):

Since lots of students mess their system now. You have to be very careful to backup your whole system before you start to make a new partition for MINIX. As I knew, most of you have some large disks whose spaces are over 512 Mbytes. This may derive serious troubles to make a new partition for any PC unix system if you have already partitioned them by dos, win95, or nt partition tools. Because all of these large size disks need to use some tools embedded in your EIDE or SCSI card to extend the partition table such that the O.S. can read the maximum size of the hard disks. For EIDE, it is LBA, for SCSI, there should have an options to enable the functions similar to the LBA. But this action actually fool the OS with the fake number of cylinders, sectors and heads. That is what the large disk space you

saw. However, for a unix system you should not enable this option. Since unix can directly recognize the real size of the disk. That is the most possible reason why students failed to install the MINIX on a new partition which was even not created by fips. The only way to do it successfully is to backup your all of stuff first, then repartition the disk for dos, win95, and MINIX by the MINIX's part tool, then re-install dos, win95, and MINIX. Now, you ought to have all of them together. Again, NT is still having problem to join with them together unless you have a commercial tools, like the sys commander. One frequent problem people encounter is that their defragmentation program leaves unmovable sectors at the end of the partition. It is addressed in detail in FIPS.DOC.

INSTALLING MINIX :

```
+-----+
| This is a log of a Minix installation on my notebook |
| in the form of a number of screen images with bits of |
| explanatory text in these boxes. The floppy images |
| have already been made and the system is booted with |
| a diskette that contains both ROOT and USR combined. |
| That is why I typed 'fd0c' after starting Minix: |
+-----+
```

Minix 2.0.0 Copyright 1997 Prentice-Hall, Inc.

Executing in 32-bit protected mode

Memory size = 8010K MINIX = 295K RAM disk = 480K Available = 7236K

RAM disk loaded.

Tue Oct 1 16:21:37 MET 1996

Finish the name of device to mount as /usr: /dev/fd0c

/dev/fd0c is read-write mounted on /usr

Starting standard daemons: update.

Login as root and run 'setup' to install Minix.

Minix Release 2.0 Version 0

noname login:

```
+-----+
| As suggested I log in as root and enter 'setup' to |
| start the installation script. |
+-----+
```

noname login: root

setup

This is the Minix installation script.

Note 1: If the screen blanks suddenly then hit F3 to select "software scrolling".

Note 2: If things go wrong then hit DEL and start over.

| This notebook has only one disk, so I type 'r': |

+-----+

```

Select device  ---first--- --geom/last-- -----sectors-----
Device        Cyl Head Sec  Cyl Head Sec   Base   Size   Kb
/dev/hd0      485  16 63
              0  0 0 488 15 62    0 492912 246456
Num Sort  Type
1 hd1 12          0  1 0  4 15 62    63   4977  2488
2* hd2 81 MINIX   5  0 0 426 14 52   5040 425303 212651
3 hd3 00 None     0  0 0  0  0 -1    0    0    0
4 hd4 00 None     0  0 0  0  0 -1    0    0    0

```

at-hd0: Conner Peripherals 240MB - CP2251

+-----+

```

| There are two partitions already present on the disk, |
| one that contains setup stuff for the notebook and a |
| 208 megabyte partition that already contains Minix. |
| The AT disk driver has tickled the IDE disk to make |
| it tell its type and make. It is a message from deep |
| in the kernel, not from part. |
| For this install I want a new partition, so I move |
| over to the Size field of the first unused slot, line |
| 3 with type "None". I could type a number now, but |
| I want to use the rest of the disk, so I hit 'm' |
| there once or twice to let part propose something. |

```

+-----+

```

Size in sectors  ---first--- --geom/last-- -----sectors-----
Device          Cyl Head Sec  Cyl Head Sec   Base   Size   Kb
/dev/hd0        485  16 63
              0  0 0 488 15 62    0 492912 246456
Num Sort  Type
1 hd1 12          0  1 0  4 15 62    63   4977  2488
2* hd2 81 MINIX   5  0 0 426 14 52   5040 425303 212651
3 hd3 81 MINIX   426 14 53 488 15 62  430343  62569  31284
4 hd4 00 None     0  0 0  0  0 -1    0    0    0

```

+-----+

```

| Note that the end of partition 2 and the start of |
| partition 3 are not on a cylinder boundary. Minix |
| doesn't care about that, but Linux fdisk would throw |
| a fit if it would see this. In this case I'm happy, |
| so I type 'q' to quit, and 'y' to save the table: |

```

+-----+

```

Size in sectors  ---first--- --geom/last-- -----sectors-----
Device          Cyl Head Sec  Cyl Head Sec   Base   Size   Kb
/dev/hd0        485  16 63
              0  0 0 488 15 62    0 492912 246456
Num Sort  Type
1 hd1 12          0  1 0  4 15 62    63   4977  2488
2* hd2 81 MINIX   5  0 0 426 14 52   5040 425303 212651
3 hd3 81 MINIX   426 14 53 488 15 62  430343  62569  31284
4 hd4 00 None     0  0 0  0  0 -1    0    0    0

```

Save partition table? (y/n) y

Please finish the name of the primary partition you have created:
(Just type RETURN if you want to rerun "part") /dev/hd3

You have created a partition named: /dev/hd3
The following subpartitions are about to be created on /dev/hd3:

```
Root subpartition: /dev/hd3a 1440 kb
/usr subpartition: /dev/hd3c rest of hd3
```

Hit return if everything looks fine, or hit DEL to bail out if you want to think it over. The next step will destroy /dev/hd3.

:

```
+-----+
| I hit ENTER:                |
+-----+
```

Migrating from floppy to disk...

Scanning /dev/hd3c for bad blocks. (Hit DEL to stop the scan if are absolutely sure that there can not be any bad blocks. Otherwise just wait.)

```
cp /fd0/etc/hostname.file /mnt/etc/hostname.file
cp /fd0/etc/motd /mnt/etc/motd
```

```
+-----+
| Lots of files are put on the disk. It is a combined |
| ROOT+USR diskette, so both parts are installed with- |
| out pause. When done I am greeted with the following |
| message, showing that Minix has a short memory, |
| because at boot it said: Memory size = 8010K. I |
| could type 8010, but ENTER will do. |
+-----+
```

```
mkdir /mnt/root
mkdir /mnt/tmp
cp /fd0/.ellepro.b1 /mnt/.ellepro.b1
/dev/fd0 unmounted from /fd0
/dev/hd3a unmounted from /mnt
```

What is the memory size of this system in kilobytes? [4096 or more]
Second level file system block cache set to 1024 kb.

```
+-----+
| Setup is now done with the migration to disk, so it |
| wants me to type 'halt' and 'boot hd3'. It says |
| "please", so I comply: |
+-----+
```

Please insert the installation ROOT floppy and type 'halt' to exit Minix. You can type 'boot hd3' to try the newly installed Minix system.

```
# halt
System Halted
fd0>boot hd3
```

Minix 2.0.0 Copyright 1997 Prentice-Hall, Inc.

Executing in 32-bit protected mode

at-hd0: Conner Peripherals 240MB - CP2251

Memory size = 8010K MINIX = 295K RAM disk = 1024K Available = 6692K

Tue Oct 1 16:39:24 MET 1996

/dev/hd3c is read-write mounted on /usr

Starting standard daemons: update.

Login as root and run 'setup /usr' to install floppy sets.

Minix Release 2.0 Version 0

noname login:

```
+-----+
| Type 'root' and 'setup /usr'. ENTER, ENTER, input |
| floppy 1 of the USR.TAZ set, and ENTER.           |
+-----+
```

noname login: root

setup /usr

What is the size of the images on the diskettes? [all]

What floppy drive to use? [0]

Please insert input volume 1 and hit return

created directory: local/lib

created directory: local/man

created directory: local/src

created directory: man

Can't make directory mdec: File exists

```
+-----+
| Usage(8) warns about this "File exists" message, and |
| that it doesn't matter. Otherwise this would go     |
| wrong:                                               |
+-----+
```

mdec/boot, 050 tape blocks

mdec/bootblock, 01 tape blocks

mdec/extboot, 01 tape blocks

```
+-----+
| Lots of files installed, I add floppy 2 and 3 of    |
| USR.TAZ when it asks for them. When done I type   |
| "setup /usr" again for SYS.TAZ, and once more for |
| CMD.TAZ. Five floppies later and I'm done.        |
+-----+
```

src/commands/talkd/talk.h, 03 tape blocks

src/commands/talkd/talkd.c, 03 tape blocks

src/commands/talkd/talkd.h, 01 tape blocks

/tmp/LAST, 0 tape blocks

Done.

```

+-----+
| As an example I give the system the name "finiah", |
| but seeing one of Michael Temari's network programs |
| extracted inspires me to make a network kernel. Log |
| in as 'bin' and bring <minix/config.h> into the one |
| true editor of the aging computer geek:           |
+-----+

```

```

# echo finiah >/etc/hostname.file
# ^D
Minix Release 2.0 Version 0

```

```

finiah login: bin
! vi /usr/include/minix/config.h

```

```

+-----+
| I change the following parameters for more processes, |
| a bigger cache, no 2nd level cache, networking, only |
| the AT/IDE disk driver and 4 pseudo ttys for remote   |
| login:                                                |
+-----+

```

```

#define NR_PROCS      40

#if (MACHINE == IBM_PC && _WORD_SIZE == 4)
#define NR_BUFS      1024 /* # blocks in the buffer cache */
#define NR_BUF_HASH  2048 /* size of buf hash table; MUST BE POWER OF 2*/
#endif

#define ENABLE_CACHE2  0

#define ENABLE_NETWORKING 1 /* enable TCP/IP code */
#define ENABLE_AT_WINI  1 /* enable AT winchester driver */
#define ENABLE_BIOS_WINI 0 /* enable BIOS winchester driver */
#define ENABLE_ESDI_WINI 0 /* enable ESDI winchester driver */
#define ENABLE_XT_WINI  0 /* enable XT winchester driver */
#define ENABLE_ADAPTEC_SCSI 0 /* enable ADAPTEC SCSI driver */

#define NR_PTYS      4 /* # pseudo terminals (0 to 64) */

```

```

+-----+
| The main kernel Makefile is in /usr/src/tools. The |
| command 'make hdbboot' compiles a new image and puts |
| it in /minix/, next to the distribution kernel:     |
+-----+

```

```

! cd tools/
! make hdbboot

```

```

+-----+

```

```

generic/tcp_send.o generic/udp.o
exec cc -c -m -l. -l. -l/usr/include putk.c
exec cc -o inet -i inet.a putk.o
install -S 10k inet
exec cc -O -D_MINIX -D_POSIX_SOURCE init.c -o init
install -S 192w init

```

```
installboot -image image ../kernel/kernel ../mm/mm ../fs/fs ../inet/inet init
text data bss size
54432 9000 38624 102056 ../kernel/kernel
12544 1176 32072 45792 ../mm/mm
28816 2196 1101676 1132688 ../fs/fs
66480 7420 94400 168300 ../inet/inet
6828 2032 1356 10216 init
-----
169100 21824 1268128 1459052 total
exec sh mkboot hdboot
cp image /dev/hd3a:/minix/2.0.0r0
Done.
```

```
+-----+
| Shut the system down to return to the boot monitor. |
| The 2nd level cache is no longer needed, so I set the |
| RAM disk size to zero. The NE2000 card of the note- |
| book is at I/O address 300, IRQ 5. Save and boot. |
+-----+
```

! shutdown

Broadcast message from bin@finiah (console)

Tue Oct 1 18:08:49 1996...

The system will shutdown NOW

```
System Halted
hd3a>unset ramsize
hd3a>DPETH0=300:5
hd3a>save
hd3a>boot
```

```
+-----+
| The new boot messages show that things have changed. |
| The ethernet driver has found an NE2000, the larger |
| block cache has made the kernel (MINIX) larger, and |
| the RAM disk is done away with. One of my other PC's |
| has seen fit to answer the RARP request (here is my |
| ethernet address, what is my IP address?) of my |
| notebook, but the notebook doesn't yet know how to |
| translate 127.36.93.27 to its own name yet. |
+-----+
```

Minix 2.0.0 Copyright 1997 Prentice-Hall, Inc.

Executing in 32-bit protected mode

```
ne2000: NE2000 at 300:5
at-hd0: Conner Peripherals 240MB - CP2251
```

Memory size = 8010K MINIX =1441K RAM disk = 0K Available = 6569K

```
Tue Oct 1 18:12:41 MET 1996
/dev/hd3c is read-write mounted on /usr
Starting standard daemons: update.
Starting network daemons: rarpd nonamed irdpd talkd.
```

Starting network services: shell login telnet ftp.

Minix Release 2.0 Version 0

127.36.93.27 login: root
shutdown -r

COMPILING KERNEL :

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.

HOW TO DO "FDBOOT" :

You should go to the tools directory by typing `cd /usr/src/tools` and with blank (formatted) floppy inside the floppy drive, type 'make fdboot' at the prompt on tools directory. That will copy your kernel image to floppy. After it finishes you should halt the system by typing 'halt' at prompt and then type 'boot fd0'. Thus the system will boot from floppy.

HINTS FOR DOING PROGRAMMING ASSIGNMENTS :

For all your assignments, you have to mount all modified source files to the bootable disk. To mount you can use the following commands :

`-mount /dev/fd0 /fd0`

`-make` sure you are in the directory where modified source files exist. If not, go there by simple `cd` command

`-copy` files you want using the `cp` command. For example – `cp proc.c /fd0`

`-After` copying all the files go to the tools directory, umount it by command,
`umount /dev/fd0`

`-You` can check the files before and after the umount, by the command
`ls /fd0`

The above commands are for making the copy of your source files on your bootable floppies. You cannot access those in DOS or Windows.

For using files to copy and paste for your documentation, you have to take another dos formatted disk and follow as below (for doswrite)

-To copy source files from system to floppy, go to the directory and give the command

```
doswrite fd0 (name of the file you want on floppy) < proc.c (or any name from the directory)
ie doswrite fd0 (name of file on floppy) < proc.c
```

- To copy your file from disk to system, go to the corresponding directory. Then do

```
dosread -a fd0 (name of the file on floppy) > (name of the file you want on the system)
```

```
ie dosread -a fd0 proc.c > proc.c
```

- You can use `dosdir -l fd0` to see your files on the floppy

Then you can access this floppy from the Windows Explorer of your machine and open it in Visual C++ (or any editor) , copy/paste it to your Word Document. This second DOS floppy is just for your use, you just have to submit MINIX bootable floppy (with source files mounted on it)

Printing

`lp test.c`

prints test.c file to your printer. type `man lp` to get more info.

Floppies

You can use dos formatted floppies to transfer files to and from MINIX using the *dosread* command :

```
dosread -a fd0 prog.c > myprog.c ( copying prog.c from floppy to current
directory as myprog.c)
```

```
doswrite fd0 test.c < prog.c ( copy prog.c from MINIX to floppy as test.c)
```

```
dosdir -l fd0 ( show the files of a dos floppy )
```

You could format your disks though with

```
format /dev/fd0 720 : formats 720 KB diskettes on floppy A
```

```
format /dev/fd0 1440 : formats 1440 KB diskettes on floppy A
```

and then use

```
mkfs /dev/fd0 : makes file system on floppy A
```

to create the minix filesystem on the floppy .

Now in order to transfer files you will have to mount the floppy on the MINIX filesystem

```
mount /dev/fd0 /fd0
```

the directory `/fd0` will contain the information on your floppy. By copying files to this directory you are essentially copying files to the disk.

Remember though to unmount your disk after you have finished copying files to your floppy media

```
umount /dev/fd0
```

Compiling

To compile files in a directory, you can do several things. Compilation of OS takes sometimes quite a long time on slower machine. `make` is the command to compile source in the system. It looks for "Makefile" and executes this file. "Makefile" simply keeps the file names and their compilation rules in text format. `make` also checks the date of files. If compiled date of a file is older than the source file or ".o" file of this file doesn't exist in the directory, it compiles the file otherwise it skips the file. You need to monitor this compilation process. Each modified file is compiled first, then linked together. If something goes wrong with this process you should delete all the ".o" files, then recompile again. If `make` can't compile, you can also check the date of files. They cannot be future dates !!!!

To compile

- ?? minix kernel, you can type under the `/usr/src/tools` directory "make all"
- ?? a minix source file, you can type "cc -c -m -I/usr/src/kernel proc.c " If your machine is slow this is the efficient way to make sure your modifications are okay. Later you can compile all files.
- ?? files in a directory, you can type "make" but "Makefile" file should exist in the directory.
- ?? compiles everything in system, including OS and programs, under the `/usr/src/tools` directory type "make world"

Processes

Using the *ps* command you can view processes that are currently running on your machine along with their PID if you need to manually terminate one of them.

Editor

VI is the MINIX editor. Basic vi commands are attached with this document

Shutdown/Reboot

You could use

```
reboot
```

or

```
shutdown -r now
```

to reboot the machine.

```
shutdown -h now
```

will shutdown the machine in order to turn it off.

Grep

If you want to find out in which other file you can find the string "*pid*" you could use :

```
grep pid *.c
```

This will look into all the files in the current directory with extension *.c* for the string *pid* and output the name of the file and the line that contains *pid*. Look up *grep* in the man pages for more information on this powerful command. See the attached vi Editor Reference Card