NEXTSTEP In Focus, Spring 1994 (Volume 4, Issue 2).

Adding On without Flipping Out

Mark Tacchi

NEXTSTEP provides features to make it as easy as possible to install a new disk on a computer. This article describes how to format, initialize, and partition SCSI and IDE disk drives, as well as how to mount them at boot time.

Using SCSI and IDE Drives

Installing a new disk on some computers can be complicated, particularly if you want to use sophisticated features like partitioning and automounting. NEXTSTEP addresses these difficulties with tools for formatting, initializing, and partitioning SCSI and IDE disk drives. It also lets you set up disks to mount automatically at boot time. If you want to install disks that conform to either the SCSI or IDE specifications, you should have little trouble adding them to your computer
in any combination.

Formatting and Initializing

Before you can use a disk with any computer, the disk must be formatted and initialized. Formatting is a low-level operation that places timing marks on the disk so that the hardware can distinguish where physical block sectors start and end. Floppy disks and other removable media with a total storage capacity of 6 megabytes or less generally aren't formatted when you

buy them. In contrast, hard disks and storage media that have more than 6 megabytes of storage capacity are usually formatted at the factory where they're made.

Initialization places an internal label and other items on the disk, such as a superblock, inodes, and other file system elements for storing, retrieving, and managing data. Generally, disks aren't initialized when you buy them.

When you connect an uninitialized or unrecognized hard disk to a computer, NEXTSTEP asks whether you want to initialize the disk or ignore it. If you choose to initialize the disk, NEXTSTEP asks you to select the disk name and file system type. It then initializes the disk with the chosen file system and mounts it. If you insert an unformatted floppy disk in the computer, NEXTSTEP follows
the same steps but automatically formats the disk before initializing and mounting it.

In the infrequent case in which you have an unformatted disk that's larger than 6 megabytes, NEXTSTEP can't automatically format it. Instead, you must format it yourself before initializing it. To do this for a SCSI disk, use the **sdform** utility. There's currently no comparable utility for IDE drives in NEXTSTEP, so you must resort to DOS utilitiesÐor if your machine comes with hard disk maintenance utilities, you can use them.

*For information on the **sdform** SCSI disk formatting utility, please see the **sdform** UNIX manual page.*

Partitioning

When NEXTSTEP automatically initializes a hard disk, it creates a single partition that occupies the entire disk. A single partition arrangement is very simple to use and requires minimal system administration. However, sometimes you might prefer to create a disk with several partitions to manage data a special way, contain disk damage, facilitate backup operations, or export multiple file systems from the same disk with different access privileges. There are several ways to partition a disk, depending on whether the disk is a SCSI or IDE disk.

Partitioning with fdisk

If you need to access operating systems in addition to NEXTSTEP at boot time, you may wish

to partition your disk with the utility **fdisk**. You can use NEXTSTEP's **fdisk** on an Intel-based computer to create up to four partitions on your drive. The command modifies the DOS partition table that resides on the boot sector of bootable disks.

To run **fdisk**, you must be **root**. You can run **fdisk** in interactive mode by specifying the raw device with no inquiry or action. Or, to run **fdisk** on the *live partition*Ðthe entire secondary IDE disk driveÐspecify the device **hd1h**:

```
# fdisk /dev/rhd1h
```

Note that the NEXTSTEP booter and **fdisk** command can identify only one physical NEXTSTEP partition.

*To find out more about **fdisk**, see the **fdisk** UNIX manual page and Divvying It Up: How to Use **fdisk**º (NEXTSTEP In Focus 3 [Spring 1993]), NeXTanswers #1131.*

BSD 4.3 UNIX partitioning

To build a disk with one or two BSD 4.3 UNIX partitions, you can use the BuildDisk application in **/NextAdmin**. You can also use this application to place the NEXTSTEP operating system on the disk.

(BSD 4.3 has a fixed partition limit of 2 gigabytes (2^{31}bytes) and drives that exceed this boundary must be partitioned.)

If you want more than two partitions or some special characteristics for your disk, you must create a **disktab** entry and use it to initialize the disk from the command line. The **/etc/disktab** file contains data that describes each disk's native geometry and how the computer should partition each disk when it initializes it. You don't need to place an entry in the **disktab** file for every disk you want to add to the systemÐthe initialization software can read the necessary data from any disk drive that supports the SCSI MODESENSE command. Rather, you provide a **disktab** entry if you want to format your disk in some special way.

To create a **disktab** entry, you must determine the geometry of your disk, choose a location and size for each partition, and then add lines describing the partitions you want into the **/etc/disktab** file.

Note: NeXT Computer, Inc., doesn't officially support modifications to the **disktab** file. You should not create **disktab** entries unless you have had extensive experience with disks in a UNIX environment. This information is provided to point experienced administrators to the right

tools.

Analyzing the disk

To determine your drive's name, geometry, and size, contact the disk vendor or use the command **scsimodes** to get this information for an existing disk from the same vendor. This utility displays information about a SCSI disk. **scsimodes** takes a single command line argument, the name of the raw disk device. You must run this command as **root**:

```
# scsimodes /dev/rsd0a
SCSI information for /dev/rsd0a
Drive type: SEAGATE ST43400N
512 bytes per sector
99 sectors per track
21 tracks per cylinder
2737 cylinder per volume (including spare cylinders)
9 spare sectors per cylinder
21 alternate tracks per volume
5688446 usable sectors on volume
```

If your system already has a labeled disk of the type you want to partition, you can get a more complete set of **disktab** information by writing a program that invokes the **getdiskbyname()** system call. Please see the **getdiskbyname** UNIX manual page for details.

*The **scsimodes** command is not officially documented nor supported by NeXT.*

*IDE drives don't respond to the **scsimodes** commandÐyou'll have to get data about any IDE drive from the disk drive vendor.*

Editing the disktab file

Before you edit the **disktab** file, please be certain that you understand its format. Review the UNIX manual page on **disktab** and the system's existing **disktab** file.

The **disktab** file contains a list of entries, one for each disk configuration. Each entry consists of a single UNIX ªlineº It typically takes several lines within a text editor to write an entryÐintermediate lines have the UNIX escape backslash character (****) before the carriage return, so that the entry is recognized as a single line.

Entries in **disktab** consist of a number of colon-separated fields. The first entry for each disk gives the names for the disk, separated by vertical bar characters (**|**). The last name listed is a

long name that fully identifies the disk.

Figure 1 shows a typical **disktab** entry for a SCSI disk. This is the entry for the 2.5 gigabyte Seagate drive described by the **scsimodes** command above. Note that because this drive is larger than 2 gigabytesÐ2^{31} bytes, or 2,147,483,648 bytesÐit must be partitioned into multiple BSD 4.3 UNIX partitions.

Figure 1: *A **disktab** entry for a Seagate SCSI disk drive*

```
ST43400N|ST43400N-512|SEAGATE ST43400N-512:\
        :ty=fixed_rw_scsi:nc#2737:nt#21:ns#99:ss#512:rm#3600:\
        :fp#160:bp#0:ng#0:gs#0:ga#0:ao#0:\
        :os=sdmach:z0#32:z1#96:hn=localhost:ro=a:\
        :pa#0:sa#4194304:ba#8192:fa#1024:ca#32:da#4096:ra#10:\
                :oa=time:ia:ta=4.3BSD:\
        :pb#4194304:sb#1494142:bb#8192:fb#1024:cb#32:db#4096:\
                :rb#10:ob=time:ib:tb=4.3BSD:
```

The name of the drive is the same as the name shown by the **scsimodes** command, with the physical sector size of the disk appended at the end. If you name a **disktab** entry according to this convention, the BuildDisk application can use the entry to format and initialize the disk. If you name it another way, you must use the **disk** command with the **-i** and **-t** options to initialize the disk according to the **disktab** entry.

Figure 2 shows what the other fields in the **disktab** entry specify.

Figure 2: *Fields in a **disktab** file*

Field	Description
ty	Type of disk. Required for all entries. Valid types are **removable_rw_scsi**, **fixed_rw_scsi**, **fixed_rw_ide**, **removable_rw_optical**, **removable_rw_floppy**.
ns	Number of sectors per track. This is a critical field, used for performance tuning at initialization.
nt	Number of tracks per cylinder. Helps decide where to put superblock backups. Not a critical field.
nc	Total number of cylinders on the disk. Not used by NEXTSTEP.
p[a-h]	Base sector numbers of partitions **a** through **h**.
s[a-h]	Sizes of partitions **a** through **h**.

b[a-h] Block sizes for partitions **a** through **h**, in bytes. This is the size of a logical block in the file system. The NEXTSTEP file system block size is 8192 bytes.

f[a-h] Fragment sizes for partitions **a** through **h**, in bytes. This is the Berkeley file system **frag**. Set it to 1024 bytes; there's no advantage to making it larger.

c[a-h] Partition "cylinders-per-group" for **newfs**. This is a "black magic" parameter for disk block managementÐuse **32**. BuildDisk or **mkfs** (which runs when you call **newfs**) lets you know if you need to change it. See **d**[a-h], partition density.

d[a-h] Partition densityÐbytes-per-inodeÐfor **newfs**. Indicates how large files in the partition will be on the average, and thus how many inodes it should provide. A good general value is **4096**. If you will have just a few large files on the disk, make this number larger; if you expect to have many very small files, specify a smaller number. If NEXTSTEP can't set up a file system with the parameters as you've set them, it will suggest that you change the cylinders-per-group parameter, **c**[a-h].

r[a-h] Partition minimum left free for **newfs**, a value from **0** to **100**. The percent of formatted disk space that should always be left free, so blocks can be distributed among files efficiently. You can save space by using a smaller minfreeÐfloppy disks have a minfree of 0% for because they're small and slow. However, a minfree less than 10% isn't usually recommended for hard disks.

o[a-h] Partition optimization for **newfs**; accepted values are **space** and **time**. Set to **time** if possible. Use **space** only when space on the disk is critically short.

i[a-h] Whether the system should run **newfs** on partition during initialization with **disk**. Include this field in the entry for any partition that will have a UNIX file system on it; omit it otherwise.

m[a-h] Partition mount point name. Not used for NEXTSTEPÐthe mount point is established by the automounter or by the entry in the local **/etc/fstab** file.

a[a-h] Partition automount on insert. Unused by NEXTSTEP.

t[a-h] Partition file system type, like **4.3BSD** or **sound**. Set to **4.3BSD**. Not used now, but may be in the future.

rm Rotational speed of the disk in rotations per minute, for performance tuning. The default value is **3600** for hard disks, **300** for floppy disks. This information isn't reported by **scsimodes**Ðobtain the value from the disk manufacturer. If the disk is the same type as one you already have in your system, you can get this value by writing a program that calls **getdiskbydev()**. If you can't find the true value, the default will suffice.

ss Sector size in bytes. Boot disks on Intel-based computers must have 512 bytes/sector. Data disks can have 1024 bytes/sector. For other architectures, both boot and data disks can have 1024 bytes/sector.

fp Number of sectors in the "front porch" of the disk. These are 1024-byte sectors; set this value to **160** for hard drives. Floppies use **92**.

bp Number of sectors in "back porch" of the disk. **0** for all current devices.

ng Number of alternate groups on the disk. Only for optical disks. Set to **0** or omit.

gs Number of sectors per alternate group. Only for optical disks. Set to **0** or omit.

ga Number of alternate sectors per alternate group. Only for optical disks. Set to **0** or omit.

ao Sector offset of alternates in alternate group. Only for optical disks. Set to **0** or omit.

os Name of file to boot from. Set to **sdmach** for NeXT computers and to **mach_kernel** for HP PA-RISC, Intel-based, and Sun SPARC computers.

z[0-1] Locations of first-level boot code. Set to **32** and **96** respectively for hard disks, **32** and **32** for floppy disks.

hn Host name to be written to label.

ro Read-only root partition. Unused by NEXTSTEP.

rw Read/write partition. Unused by NEXTSTEP.

For the sector fields, if you plan to have 1024-byte sectors on a 512-byte/sector disk, divide the value by 2 and round up. You can't use 1024 bytes per sector for boot disks on Intel-based computers.

newfs *only sets up the file system, whereas BuildDisk attempts to format, initialize, and build a bootable disk. Use* ***newfs*** *to create an unbootable data disk.*

Initializing the disk

When you've created the entry in the **disktab** file, you can build the disk with BuildDisk or simply initialize it with the **disk** command. If the **disktab** file contains an entry for a disk and if the entry name matches the name of the disk, BuildDisk uses the **disktab** entry.

```
# disk /dev/rsd0a
```

You can use **disk** interactively if the name of the drive matches the name of the disktab entry. The **init** command initializes the disk. For help in **disk**'s interactive mode, type **help** at the **disk** prompt. Otherwise you can enter the **disktab** entry name after the **-t** flag:

```
# disk -i -t ST43400N /dev/rsd0a
```

Unlike with other UNIX and disk systems, you don't have to start a partition on a track or cylinder boundary. However, if you don't start the partition on a boundary, you might waste a little space. Check for messages from **mkfs** in the Console.

Mounting Drives and Partitions

If there is no entry in **/etc/fstab** for a disk or partition, the disk or partition won't be mounted until a user logs in. Once someone logs in and Workspace Manager starts up, NEXTSTEP examines all unmounted devices to see whether they've been initialized for a file system supported by NEXTSTEP. Supported file systems include Berkeley UNIX 4.3, CD-ROM, DOS, Macintosh, and audio compact disc.

If a drive conforms to one of these standard filesystems, Workspace Manager sends a mount request to the drivers. NEXTSTEP mounts the drive on the root directory of the filesystem as */disklabelname*. Drives and partitions mounted in this manner are owned by the current user. In contrast, drives and partitions without entries in **/etc/fstab** are owned by the current user when mounted.

Drives and partitions with /etc/fstab entries

NEXTSTEP mounts any connected drive or partition that has an entry in **/etc/fstab** when the computer boots. By default, such disks and partitions are owned by **root**. The system disk, which holds the root file system, always has an entry in **/etc/fstab**, placed there when NEXTSTEP was installed.

Additional partitions on the system disk or any other disks must be mounted manually or be given an **/etc/fstab** entry. The automounter sees only the first partition on a disk and partitions that have entries in the **/etc/fstab** file.

Conversely, NEXTSTEP will try to mount every disk it sees, either at boot time or when the Workspace starts up, unless there is an **/etc/fstab** entry for it that includes the **-noauto** option. Please see the UNIX manual page on **fstab** for more information.

Creating /etc/fstab entries

Figure 3 shows three examples of **/etc/fstab** files. The first has entries for two SCSI drives, the second for two IDE drives, and the third for one SCSI drive and one IDE drive.

Figure 3: *Entries from **/etc/fstab** files for different disk drive configurations*

SCSI only:

```
/dev/sd0a / 4.3 rw,noquota,noauto 0 1
/dev/sd1a /home 4.3 rw,noquota 0 2
/dev/sd1b /data 4.3 rw,noquota 0 2
```

IDE only:

```
/dev/hd0a / 4.3 rw,noquota,noauto 0 1
/dev/hd1a /home 4.3 rw,noquota 0 2
```

SCSI and IDE:

```
/dev/hd0a / 4.3 rw,noquota,noauto 0 1
/dev/sd0a /SCSI_Disk 4.3 rw,noquota 1 2
/dev/sd1a /SCSI_CDROM 4.3 ro,noquota,removable 1 2
/dev/hd1a /IDE_Disk 4.3 rw,noquota 1 2
```

You can display these entries by executing **mount -p** once all devices are mounted; devices that have entries in the mount table, **/etc/mtab**, are displayed. These are formatted to match that required for **/etc/fstab** and can be directly appended.

Figure 4: *Output from **mount -p***

```
# mount -p | grep dev
/dev/hd0a   /   4.3    rw,noquota,noauto   0 1
/dev/sd0a   /NEXTSTEP_3.2   4.3   ro,noquota,removable   1 2
/dev/sd1a   /Seagate SCSI disk   4.3   rw,noquota   1 2
/dev/hd1a   /Secondary_IDE   4.3   rw,noquota  1 2
```

Although DOS partitions may appear in this output, you can't mount DOS partitions at boot time in this manner. DOS partitions require special treatmentÐsee the next section.

DOS partitions

DOS partitions are automatically mounted when a user logs into the Workspace. However, you can force the DOS partition to be mounted at boot time, so that remote users and anyone who logs in without the Workspace can access the partition.

To automount the DOS partition at boot time, follow these steps:

1. Append these lines to **/etc/rc.local**. Substitute the name of the device you want to mount for **/dev/rhd0h**.

   ```
   kl_util -l dosfs
   ```

```
mount -t dos /dev/rhd0h /DOS
```

In the **mount** command, **-t dos** indicates you're mounting a DOS file system. To mount another device such as the secondary IDE drive, use **/dev/rhd1h** as the raw device.

2. Append this line to **/etc/kern_loader.conf**:

```
/usr/filesystems/DOS.fs/dosfs_reloc
```

3. Reboot.

Note that **root** doesn't own this mount; the mount and its subdirectories and files are owned by the current user. This is a security hazard, but it's a DOS-ism that has existed since DOS' inception.

Conclusion

You should now have enough information on formatting, initializing, partitioning, and mounting disks to play with them on your own. If you run into any obstacles, refer to the UNIX manual pages for specifics.

I'll leave you with this final cautionary note: Back up valuable data before experimenting, because an honest mistake as **root** can be destructive to dataÐand to your day!

*Mark Tacchi is a member of the Technical Support Team. You can reach him by e-mail at **tacchi@next.com**.*
Flip Dibner contributed material on SCSI devices for this article.

*What did you think of this article? Please send feedback to **journal_info@next.com** to let us know whether this article was useful and to tell us how we can make this journal a great resource for you!*