

# Adding a New Hard Disk

Aeleen Frisch

I've been a system administrator for more than 20 years. While many aspects of the job have stayed pretty much the same, some tasks are now very different from what they previously were. One of the latter is administering disk space. In the past, one had to use a lot of tricks to efficiently manage what was then a very precious commodity. Now however, with the advent of relatively cheap disks, having enough storage space is rarely an issue since adding more disk space is seldom a problem.

I've been a system administrator for more than 20 years. While many aspects of the job have stayed pretty much the same, some tasks are now very different from what they previously were. One of the latter is administering disk space. In the past, one had to use a lot of tricks to efficiently manage what was then a very precious commodity. Now however, with the advent of relatively cheap disks, having enough storage space is rarely an issue since adding more disk space is seldom a problem.

This month, we will take an in-depth look at the process of adding an additional disk drive to your system. We'll begin by summarizing the steps that are involved:

1. Install the new disk controller (if applicable). Make sure that you have kernel support for the controller (usually via the appropriate kernel module).
2. Physically attach the disk to the computer system.
3. Determine (or create) the special files that are associated with the new partitions.
4. Partition the new disk as desired.
5. Create filesystems on all of the new partitions.
6. Check the new filesystems for any consistency errors. This may not be necessary with certain filesystem types.
7. Mount the new filesystem.
8. Specify the new filesystem's boot-time behavior as well as other options in the filesystem configuration file (`/etc/fstab`).
9. Perform the appropriate ongoing filesystem management.

In the following sections, we'll consider each of the above items in turn.

## Installing the Hardware

In some cases, you will need to install an additional disk controller in order to add more disk space. This situation may come up when the existing controllers are fully populated with disks or when you want to add a different controller type to the system (e.g., SCSI to an EIDE-based system). Once the controller is installed, you can install the disk(s) as well.

If you are adding a new type or model of controller, you will need to ensure that you have kernel support for the new device type. You may choose to build a new kernel with integrated support, or, more commonly, use a loadable module that supports that device type. See the appropriate HOWTO document to determine the module that corresponds to your particular device.

Once you have set up the kernel appropriately, reboot the system.

# Adding a New Hard Disk

Aeleen Frisch

## Locating Special Files and Partitioning the Disk

Before you can do anything with a new disk, you first must determine which special file to use when referring to it. This is easy to figure out. One way is to examine the file `/proc/partitions`, which contains a list of all disks and partitions recognized by the system, as in the following example:

```
# cat /proc/partitions
major  minor  #blocks  name
22     0      241600   hdc
33     0      3153024  hde
33     1      2354656  hde1
33     64     3153024  hdf
33     65     5512     hdf1
33     66     1536192  hdf2
33     67     1536192  hdf3
33     68     75096    hdf4
```

This system has three hard disks (the items listed without partition numbers). The first two disks each have one physical partition, and the third disk has four partitions. (Note that the output generated from this command will vary from one Linux distribution to another.)

Alternatively, you can search the output of the `dmesg` command to determine the names of the disks that are recognized by your system, as shown in Listing One.

### Listing One: `dmesg` Command Output

```
# dmesg | egrep `^(s|h)d`
hdc: Pioneer DVD-ROM ATAPI Model DVD-115 0111, ATAPI CD/DVD-ROM drive
hdd: IOMEGA ZIP 250 ATAPI, ATAPI FLOPPY drive
hde: QUANTUM FIREBALL EX3.2A, ATA DISK drive

hdf: QUANTUM FIREBALL EX3.2A, ATA DISK drive
```

(If you are thinking that the disk naming on this example system is very weird, you are correct, but that's another story.)

Once you have correctly identified the disk, you must make sure that the special file corresponding to it is present in `/dev`. Rarely, you will need to create the device special files for the disk. If necessary, you can do so with the `MAKEDEV` script found in the `/dev` directory, as in this example, which would create special files for the sixth SCSI disk:

```
# cd /dev; MAKEDEV sdf
```

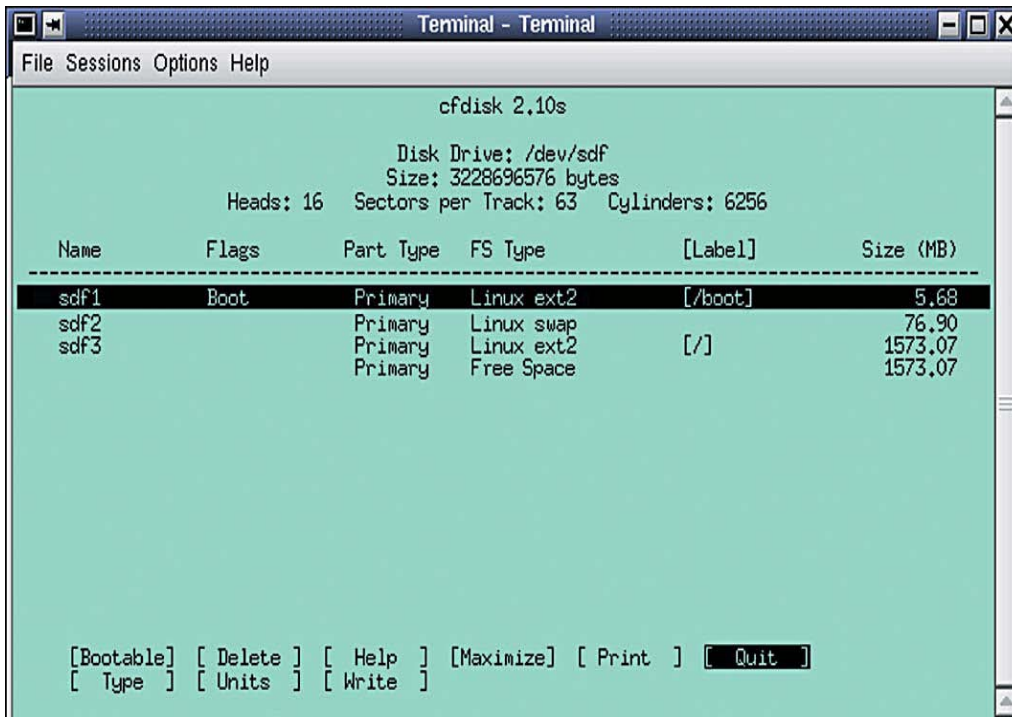
The next step is to create partitions on the new disk. Linux provides several commands for doing so: `fdisk` (the prompt-based standard utility), `sfdisk` (a partitioning tools for experts), and `cfdisk` (a pseudo-graphical partitioning tool). We'll use the latter, using the SCSI disk just mentioned as the example disk:

```
# cfdisk /dev/sdf
```

The `cfdisk` utility is illustrated in Figure One (which shows the display for a partially-partitioned disk).

# Adding a New Hard Disk

Aeleen Frisch



**Figure One: The Cfdisk Disk Partitioning Utility**

The cfdisk main window displays the disk attributes followed by a table listing the various partitions on the disk. The table's columns show many useful pieces of information: the partition's special file, any flags set for it, the kind of partition (primary or logical/extended), the partition's use, the filesystem type and mount point (if any), and the partition size in megabytes (although you can use the Units command to change the display units if desired).

For disks that have unpartitioned space available, the final entry in the partition list will be one with its FS Type field set to "Free Space," as in our example. If you select this item, one of the available cfdisk commands is New, which creates a new partition (the utility will prompt you to input additional required data such as the partition size).

After a new partition has been created, you can use other cfdisk commands to specify additional characteristics of partition. The most commonly needed of these is the Type command, which sets the partition type. Invoking this command brings up a list of all supported types. Generally, you will want to select type 82 (Linux swap) or 83 (normal Linux filesystem).

Once you have finished manipulating the partition table with cfdisk, use the Write command to save the updated table to disk. Alternatively, exiting from the utility with Quit will discard any changes that you may have made.

## Creating a Filesystem

Once partitioning is complete, the next step is to create a filesystem on the desired partition. The command to do so depends on the type of filesystem you want to create. For example, you use the mke2fs command to create an ext2 filesystem and the mkreiserfs command to create a Reiser filesystem.

# Adding a New Hard Disk

Aeleen Frisch

In most cases, all you need to do is specify the desired partition as the command's argument, as in these two examples:

```
# mke2fs /dev/sdf4
# mkreiserfs /dev/sdf4
```

## Listing Two displays some output from mke2fs.

```
# mke2fs /dev/hdf3
mke2fs 1.19, 13-Jul-2000 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
192384 inodes, 384048 blocks
19202 blocks (5.00%) reserved for the super user
...
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

hdf: QUANTUM FIREBALL EX3.2A, ATA DISK drive
```

The output lists various characteristics of the filesystem. Of these, the “superblock backups” list is the most important. These alternate superblocks can potentially allow you to recover some or all of the data on the disk should the primary superblock ever become unusable. It is always a good idea to record these values on paper or to store them in a file on a different partition or a different computer. Obviously, keeping them in a file that is on the same partition will not be much help should a failure occur.

The output from mkreiserfs is similar; it too lists a variety of filesystem attributes, including the size of the filesystem journal.

The mke2fs command has many options available that are useful in some circumstances. Table One lists some of these.

**Table One: mke2fs Command Options**

Option	Description
-b <i>size</i>	Filesystem block size — the unit in which disk space is allocated — specified in bytes (the default is 1024).
-c	Check for bad blocks before creating the filesystem.
-m <i>percent</i>	Percentage of filesystem space to reserve (it is accessible only by root). The default is 5 percent.
-i <i>ratio</i>	Bytes/inode value: create one inode for each chunk of this many bytes. The default value of 4096 usually creates more than you'll ever need, but probably isn't worth changing as the disk space savings of doing so are trivial. Alternatively, you can use -N number to specify an explicit number of inodes to create.
-v -q	Verbose or quiet output.

# Adding a New Hard Disk

Aeleen Frisch

Here is a mke2fs command that creates a filesystem on a disk partition that has been designed to be used for large scratch files:

```
# mke2fs -c -q -m 3 -N 1000 /dev/hdc1
```

This command creates a filesystem on the first partition on the third IDE disk after checking it for any bad blocks, reserving only 3 percent of the space (-m), and creating only 1000 inodes (-N). This last option makes sense in this special case because there will never be more than 100 files and directories on the filesystem. As it does its work, the mke2fs command will produce only minimal output (-q).

## Checking the New Filesystem

For ext2 filesystems, the next step in the process is to check the new filesystem with fsck:

```
# fsck -f -y /dev/sdf3
```

Table Two summarizes the most useful fsck options.

**Table Two: fsck Command Options**

Option	Meaning
-f	Force a check even if the filesystem does not appear to need one (by default, filesystem check only when they are "dirty": when they have not been dismounted properly or a certain period of time or number of system reboots has passed).
-p	Preen the filesystem, automatically fixing all problems that can be safely corrected without any data loss.
-y	Answer yes to all prompts. This has the effect of automatically fixing all problems that are found, even ones that may potentially produce a loss of data.
-b n	Use the specified backup superblock.

There is no need to check a Reiser filesystem.

## Mounting the Filesystem

The next step is to add the new filesystem to the filesystem configuration file, /etc/fstab. Figure Two contains some sample entries.

**Figure Two: An Example /etc/fstab**

```
# mount FS mount dump fsck
# device point type options freq pass
#-----
/dev/sdf3 /data1 reiserfs defaults 1 2

/dev/sdf4 /data2 ext2 noauto,user,grpид,exec 1 2
```

The fields hold: the special file name, the mount point, the type of filesystem, any options to be passed to the mount command, the dump command backup frequency, and fsck pass number. The

## Adding a New Hard Disk

Aeleen Frisch

first entry specifies a Reiser filesystem located on /dev/sdf3 to be mounted at /data1 with default options. These options are listed in Table Three.

**Table Three: Default mount Options**

Option	Meaning (opposite)
rw	Filesystem is read-write (ro).
auto	Include this filesystem when using mount -a (noauto).
nouser	Don't allow ordinary users to mount this filesystem (user).
suid	Respect SETUID/SETGID settings (nosuid).
exec	Allow execution of binary files (noexec).
nogrpuid	Use System V group ownership rules unless directory's SETGID bit is set (grpuid).
dev	Interpret corresponding device files (nodev).
aync	Perform I/O asynchronously (sync).
atime	Update file access time (noatime).

The second example entry in Figure Two corresponds to an ext2 filesystem that is located on /dev/sdf4 and which has been mounted at /data2. It will not be mounted when mount-a commands are executed (for example, at boot time); ordinary users can mount the filesystem, and the filesystem uses BSD-style group ownership rules in which new files always take their group owner from that of the directory in which they reside.

The user option implies noexec, nosuid,nodev, so this entry includes exec in order to allow binary files residing in this filesystem to be executed.

Once the /etc/fstabentries have been set up, the new filesystem may be mounted as usual with the normal mount command.

### Normal Filesystem Administration

At this point, the new filesystem will be ready for use. Of course, you will need to monitor it on an ongoing basis: make sure it doesn't get too full, back up important data, possibly defragment it from time to time, and so on.

For ext2 filesystems, you may also want to modify filesystem characteristics at some point. You can use the tune2fs command to display the status of (-loption) or to reconfigure an ext2 filesystem. You accomplish the latter using options to tune2fs. For example, the following command disables the time-between-checks function by setting the number of seconds between checks to 0 (-i) and setting the maximum number of mounts between checks to 999 (-c):

```
# tune2fs -i 0 -c 999 /dev/hdb5
Setting maximal mount count to 999
Setting interval between check 0 seconds
```

By default, Linux checks ext2 filesystems every 20 system reboots and every 30 days (whether they need to be checked or not). Thus, this command effectively disables the fsck command unless the filesystem has not been unmounted properly.

## Adding a New Hard Disk

Aeleen Frisch

tune2fs's most useful options are listed in Table Four.

**Table Four: Useful tune2fs Options**

Option	Meaning
-l	List filesystem attributes.
-cn	Maximum number of mounts between <i>fsck</i> checks.
-in	Maximum time between <i>fsck</i> checks — follow the number with d, w or m (for days, weeks or months).
-Cn	Set the mount count to this value.
-eerr	How to handle disk errors, where <i>err</i> is one of continue (ignore), remount-ro (remount the filesystem read-only), or panic (generate a kernel panic).
-m%	Percentage of disk space to reserve for root.
-rb	Number of disk blocks to reserve for root.
-uuid	User who is allowed to access the reserved space (default is 0).
-ggid	Group whose members are allowed to access the reserved space (default is 0).

### Scratching the Surface

Of course, we've only really scratched the surface of filesystem management. There is a great deal of additional information available on *mk2efs*, *fsck*, *tune2fs*, and all of the other commands we've looked at in this column; just check out the relevant man pages for each.

And finally, even though this goes without saying, we'll say it anyway — if you're going to mess around with your filesystems, make sure you back everything up first.