16

Administration Tasks

The system administrator has many responsibilities. This chapter discusses tasks not covered in Chapter 11, including configuring user and group accounts, backing up files, scheduling tasks, general problem solving, and using the system log daemon, **rsyslogd**. The chapter concludes with a section on installing and using MySQL.

In This Chapter

Configuring User and Group Accounts 556
Backing Up Files 558
System Reports 566
parted: Reports on and Partitions a Hard Disk 568
Solving Problems 574
Speeding Up the System 575
Keeping the System Secure 577
logrotate: Manages Log Files 579
Disk Quota System 582
rsyslogd: Logs System Messages 582
MySQL 584

CONFIGURING USER AND GROUP ACCOUNTS

More than a username is required for a user to be able to log in and use a system. A user must have the necessary files, directories, permissions, and usually a password to log in. At a minimum a user must have an entry in the **/etc/passwd** and **/etc/shadow** files and a home directory. The following sections describe several ways you can work with user accounts. Refer to page 387 and the *NIS-HOWTO* when you want to run NIS to manage the **passwd** database.

system-config-users: MANAGES USER ACCOUNTS

The system-config-users utility displays the User Manager window and enables you to add, delete, and modify system users and groups. To display the User Manager window, enter **system-config-users** on a command line or select **Main menu: System⇨ Administration⇨Users and Groups**. This window has two tabs: Users and Groups, where each tab displays information appropriate to its name. Figure 16-1 shows the Users tab.

Search filter

The Search filter, located just below the toolbar, selects users or groups whose names match the string, which can include wildcards, that you enter in the Search filter text box. The string matches the beginning of a name. For example, ***nob** matches **nobody** and **nfsnobody**, whereas **nob** matches only **nobody**. After you enter the string, click **Apply filter** or press RETURN. If you have only a few users, you will not need to use the Search filter.

Adding a user

To create a new user, click the **Add User** button on the toolbar. The User Manager displays the Create New User window, which gathers much of the same information as the User Data tab of the User Properties window (Figure 16-2). Enter the information for the new user and click **OK**. Once you create a user, you can modify the user to add/change/remove information.

Modifying a user

To modify a user, highlight the user in the User Manager window and click **Properties** on the toolbar; the utility displays the User Properties window (Figure 16-2). The

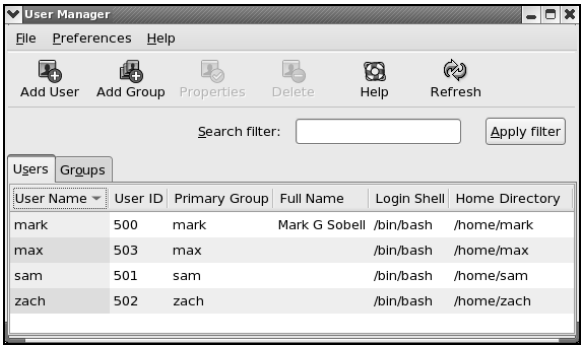

Figure 16-1 The User Manager window, Users tab

User Properties window has four tabs: User Data, Account Info, Password Info, and Groups. The User Data tab holds basic user information such as name and password. The Account Info tab allows you to specify an expiration date for the account and to lock the account so the user cannot log in. The Password Info tab allows you to turn on password expiration and specify various related parameters. In the Groups tab, you can specify the groups that the user is a member of.

Working with groups

Click the **Groups** tab in the User Manager window to work with groups. To create a group, click **Add Group** on the toolbar and specify the name of the group. To change the name of a group or to add or remove users from a group, highlight the group and click **Properties** on the toolbar. Click the appropriate tab, make the changes you want, and click **OK**. See page 472 for more information on groups.

Help

The User Manager provides extensive help. To access it, click **Help** on the toolbar.

When you are done working with users and groups, close the window.

useradd: ADDS A USER ACCOUNT

The useradd utility (and the link to it, named adduser) adds a new user account to the system. By default, useradd assigns the next highest unused user ID to a new account and specifies bash as the user's login shell. The following example creates the user's home directory (in **/home**), specifies the user's group ID, and puts the user's full name in the comment field:

```
# useradd -g 500 -c "Alex Watson" alex
```

Based on the **/etc/login.defs** file, the system creates a home directory for the new user. When useradd creates a home directory, it copies the contents of **/etc/skel**, which contains bash and other startup files, to that directory. For more information on adding and modifying user information, see the useradd and usermod man pages. Once you have added a user, use passwd to give the user a password.

Figure 16-2 The User Properties window, User Data tab

userdel: REMOVES A USER ACCOUNT

If appropriate, back up the files belonging to the user before deleting them. The userdel utility deletes user accounts. The following command removes **alex**'s account and his home directory hierarchy:

```
# userdel -r alex
```

To turn off a user's account temporarily, you can use usermod to change the expiration date for the account. Because it specifies that his account expired in the past (December 31, 2009), the following command line prevents **alex** from logging in:

```
# usermod -e "12/31/09" alex
```

groupadd: ADDS A GROUP

Just as useradd adds a new user to the system, groupadd adds a new group by adding an entry for it in **/etc/group** (page 472). The following example creates a new group named **rtfm**:

```
# groupadd -g 1024 rtfm
```

Unless you use the **–g** option to assign a group ID, the system picks the next available sequential number greater than 500. The **–o** option allows the group ID to be nonunique if you want to have multiple names for the same group ID.

The analogue of userdel for groups is groupdel, which takes a group name as an argument. You can also use groupmod to change the name or group ID of a group, as in the following examples:

```
# groupmod -g 1025 rtfm
# groupmod -n manuals rtfm
```

The first example gives the previously created **rtfm** group a new group ID number. The second example renames the **rtfm** group **manuals**.

Group ID cautions

caution The groupmod utility does not change group numbers in **/etc/passwd** when you renumber a group. You must edit **/etc/passwd** and change the entries yourself. If you change the number of a group, files that are associated with the group will no longer be associated with the group. Instead, they may be associated with no group or with another group with the old group ID number.

BACKING UP FILES

One of the most neglected tasks of system administration is making backup copies of files on a regular basis. The backup copies are vital in three instances: when the system malfunctions and files are lost, when a catastrophic disaster (fire, earthquake, and so on) occurs, and when a user or the system administrator deletes or corrupts a file by accident. Even when you set up RAID (page 37), you still need to

back up files. Although RAID provides fault tolerance (helpful in the event of disk failure), it does not help when a catastrophic disaster occurs or when a file is corrupted or accidentally removed. It is a good idea to have a written backup policy and to keep copies of backups offsite (in another building, at home, or at a completely different facility or campus) in a fireproof vault or safe.

The time to start thinking about backups is when you partition the disk. Refer to "Setting Up the Hard Disk" on page 30. Make sure the capacity of the backup device and your partition sizes are comparable. Although you can back up a partition onto multiple volumes, it is easier not to and much easier to restore data from a single volume.

You must back up filesystems on a regular basis. Backup files are usually kept on magnetic tape or some other removable media. Exactly how often you should back up which files depends on the system and your needs. Use this criterion when determining a backup schedule: If the system crashes, how much work are you willing to lose? Ideally you would back up all files on the system every few minutes so you would never lose more than a few minutes of work.

Of course, there is a tradeoff: How often are you willing to back up the files? The backup procedure typically slows down the system for other users, takes a certain amount of your time, and requires that you have and store the media (tape or disk) holding the backup. Avoid backing up an active filesystem; the results may be inconsistent, and restoring from the backup may be impossible. This requirement is a function of the backup program and the filesystem you are backing up.

Another question is when to run the backup. Unless you plan to kick users off and bring the system down to single-user mode (not a very user-friendly practice), you want to perform this task when the machine is at its quietest. Depending on the use of the system, sometime in the middle of the night can work well. Then the backup is least likely to affect users, and the files are not likely to change as they are being read for backup.

A *full* backup makes copies of all files, regardless of when they were created or accessed. An *incremental* backup makes copies of those files that have been created or modified since the last (usually full) backup.

The more people using the system, the more often you should back up the filesystems. One popular schedule is to perform an incremental backup one or two times a day and a full backup one or two times a week.

CHOOSING A BACKUP MEDIUM

If the local system is connected to a network, you can write your backups to a tape drive on another system. This technique is often used with networked computers to avoid the cost of having a tape drive on each computer in the network and to simplify management of backing up many computers in a network. Most likely you want to use a tape system for backups. Because tape drives hold many gigabytes of data, using tape simplifies the task of backing up the system, making it more likely

that you will take care of this important task regularly. Other options for holding backups are writable CDs, DVDs, and removable hard disks. These devices, although not as cost-effective or able to store as much information as tape systems, offer convenience and improved performance over using tapes.

BACKUP UTILITIES

A number of utilities help you back up the system, and most work with any media. Most Linux backup utilities are based on one of the archive programs—tar or cpio—and augment these basic programs with bookkeeping support for managing backups conveniently.

You can use any of the tar, cpio, or dump/restore utilities to construct full or partial backups of the system. Each utility constructs a large file that contains, or archives, other files. In addition to file contents, an archive includes header information for each file it holds. This header information can be used when extracting files from the archive to restore file permissions and modification dates. An archive file can be saved to disk, written to tape, or shipped across the network while it is being created.

In addition to helping you back up the system, these programs offer a convenient way to bundle files for distribution to other sites. The tar program is often used for this purpose, and some software packages available on the Internet are bundled as tar archive files.

The amanda utility (Advanced Maryland Automatic Network Disk Archiver; www.amanda.org), one of the more popular backup systems, uses dump or tar and takes advantage of Samba to back up Windows systems. The amanda utility backs up a LAN of heterogeneous hosts to a single tape drive. You can use yum to install amanda; refer to the amanda man page for details.

tar: ARCHIVES FILES

The tar (tape archive) utility stores and retrieves files from an archive and can compress the archive to conserve space. If you do not specify an archive device, tar uses standard output and standard input. With the **–f** option, tar uses the argument to **–f** as the name of the archive device. You can use this option to refer to a device on another system on the network. Although tar has many options, you need only a few in most situations. The following command displays a complete list of options:

```
# tar --help | less
```

Most options for tar can be given either in a short form (a single letter) or as a descriptive word. Descriptive-word options are preceded by two hyphens, as in **––help**. Single-letter options can be combined into a single command-line argument and do not need to be preceded by a hyphen (for consistency with other utilities, it is good practice to use the hyphen anyway).

Although the following two commands look quite different, they specify the same tar options in the same order. The first version combines single-letter options into a

single command-line argument; the second version uses descriptive words for the same options:

```
# tar -ztvf /dev/st0
# tar --gzip --list --verbose --file /dev/st0
```

Both commands tell tar to generate a (**v, verbose**) table of contents (**t, list**) from the tape on **/dev/st0** (**f, file**), using gzip (**z, gzip**) to decompress the files. Unlike the original UNIX tar utility, the GNU version strips the leading **/** from absolute pathnames.

The options in Table 16-1 tell the tar program what to do. You must include exactly one of these options in a tar command.

Table 16-1 The tar utility

Option	Effect
--append (–r)	Appends files to an archive
--catenate (–A)	Adds one or more archives to the end of an existing archive
--create (–c)	Creates a new archive
--delete	Deletes files in an archive (not on tapes)
--diff (–d)	Compares files in an archive with disk files
--extract (–x)	Extracts files from an archive
--help	Displays a help list of tar options
--list (–t)	Lists the files in an archive
--update (–u)	Like the **–r** option, but the file is not appended if a newer version is already in the archive

The **–c**, **–t**, and **–x** options are used most frequently. You can use many other options to change how tar operates. The **–j** option, for example, compresses or decompresses the file by filtering it through bzip2 (page 162).

cpio: ARCHIVES FILES

The cpio (copy in/out) program is similar to tar but can use archive files in a variety of formats, including the one used by tar. Normally cpio reads the names of the files to insert into the archive from standard input and produces the archive file as standard output. When extracting files from an archive, cpio reads the archive as standard input.

As with tar, some options can be given in both a short, single-letter form and a more descriptive word form. However, unlike tar, the syntax of the two forms differs when the option must be followed by additional information. In the short form, you

must include a SPACE between the option and the additional information; with the word form, you must separate the two with an equal sign and no SPACEs.

Running cpio with **––help** displays a full list of options.

PERFORMING A SIMPLE BACKUP

When you prepare to make a major change to a system, such as replacing a disk drive or updating the Linux kernel, it is a good idea to archive some or all of the files so you can restore any that become damaged if something goes wrong. For this type of backup, tar or cpio works well. For example, if you have a SCSI tape drive as device **/dev/st0** that is capable of holding all the files on a single tape, you can use the following commands to construct a backup tape of the entire system:

```
# cd /
# tar -cf /dev/st0 .
```

All of the commands in this section start by using cd to change to the root directory so you are sure to back up the entire system. The tar command then creates an archive (**c**) on the device **/dev/st0** (**f**). If you would like to compress the archive, replace the preceding tar command with the following command, which uses **j** to call bzip2:

```
# tar -cjf /dev/st0 .
```

You can back up the system with a combination of find and cpio. The following commands create an output file and set the I/O block size to 5120 bytes (the default is 512 bytes):

```
# cd /
# find . -depth | cpio -oB > /dev/st0
```

The next command restores the files in the **/home** directory from the preceding backup. The options extract files from an archive (**–i**) in verbose mode, keeping the modification times and creating directories as needed.

```
# cd /
# cpio -ivmd /home/\* < /dev/st0
```

Exclude some directories from a backup

tip In practice, you will likely want to exclude some directories from the backup process. For example, not backing up **/tmp** or **/var/tmp** (or its link, **/usr/tmp**) can save room in the archive. Also, do not back up the files in **/proc**. Because the **/proc** filesystem is not a disk filesystem but rather a way for the Linux kernel to provide information about the operating system and system memory, you need not back up **/proc**; you cannot restore it later. You do not need to back up filesystems that are mounted from disks on other systems in the network. Do not back up FIFOs; the results are unpredictable. If you plan on using a simple method, similar to those just discussed, create a file naming the directories to exclude from the backup, and use the appropriate option with the archive program to read the file.

Although all of the archive programs work well for such simple backups, utilities such as amanda provide more sophisticated backup and restore systems. For example, to determine whether a file is in an archive, you must read the entire archive. If the archive is split across several tapes, this process is particularly tiresome. More sophisticated utilities, including amanda, assist you in several ways, including keeping a table of contents of the files in a backup.

dump, restore: BACK UP AND RESTORE FILESYSTEMS

The dump utility, which first appeared in UNIX version 6, backs up either an entire filesystem or only those files that have changed since the last dump. The restore utility restores an entire filesystem, an individual file, or a directory hierarchy. You will get the best results if you perform a backup on a quiescent system so that the files are not changing as you make the backup.

The next command backs up all files (including directories and special files) on the **root** (**/**) partition to SCSI tape 0. Frequently there is a link to the active tape drive, named **/dev/tape**, which you can use in place of the actual entry in the **/dev** directory.

```
# dump -0uf /dev/st0 /
```

The option specifies that the entire filesystem is to be backed up (a full backup). There are ten dump levels: 0–9. Zero is the highest (most complete) level and always backs up the entire filesystem. Each additional level is incremental with respect to the level above it. For example, 1 is incremental to 0 and backs up only files that have changed since the last level 0 dump; 2 is incremental to 1 and backs up only files that have changed since the last level 1 dump; and so on. You can construct a very flexible schedule using this scheme. You do not need to use sequential numbers for backup levels. You can perform a level 0 dump, followed by level 2 and 5 dumps.

The **u** option updates the **/etc/dumpdates** file (page 471) with filesystem, date, and dump level information for use by the next incremental dump. The **f** option and its argument write the backup to the device named **/dev/st0**.

The following command makes a partial backup containing all files that have changed since the last level 0 dump. The first argument is a 1, specifying a level 1 dump:

```
# dump -1uf /dev/st0 /
```

To restore an entire filesystem from a tape, first restore the most recent complete (level 0) backup. Perform this operation carefully because restore can overwrite the existing filesystem. When you are logged in as Superuser, cd to the directory the filesystem is mounted on and give this command:

```
# restore -if /dev/st0
```

The **i** option invokes an interactive mode that allows you to choose which files and directories to restore. As with dump, the **f** option specifies the name of the device that the backup medium is mounted on. When restore finishes, load the next lower-level (higher-number) dump tape and issue the same restore command. If multiple

incremental dumps have been made at a particular level, always restore with the most recent one. You do not need to invoke restore with special arguments to restore an incremental dump; it will restore whatever appears on the tape.

You can also use restore to extract individual files from a tape by using the **x** option and specifying the filenames on the command line. Whenever you restore a file, the restored file will appear in the working directory. Before restoring files, make sure you are working in the correct directory. The following commands restore the **/etc/nsswitch.conf** file from the tape on **/dev/st0**. The filename of the dumped file does not begin with **/** because all dumped pathnames are relative to the filesystem that you dumped—in this case **/**. Because the restore command is given from the **/** directory, the file will be restored to its original location of **/etc/nsswitch.conf**:

```
# cd /
# restore -xf /dev/st0 etc/nsswitch.conf
```

If you use the **x** option without specifying a file or directory name to extract, the entire dumped filesystem is extracted. Use the **r** option to restore an entire filesystem without using the interactive interface. The following command restores the filesystem from the tape on **/dev/st0** to the working directory without interaction:

```
# restore -rf /dev/st0
```

You can also use dump and restore to access a tape drive on another system. Specify the file/directory as ***host*:*file***, where ***host*** is the hostname of the system the tape drive is on and ***file*** is the file/directory you want to dump/restore.

Occasionally, restore may prompt you with the following message:

```
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #:
```

Enter **1** (one) in response to this prompt. If the filesystem spans more than one tape or disk, this prompt allows you to switch tapes.

At the end of the dump, you will receive another prompt:

```
set owner/mode for '.'? [yn]
```

Answer **y** to this prompt when you are restoring entire filesystems or files that have been accidentally removed. Doing so will restore the appropriate permissions to the files and directories being restored. Answer **n** if you are restoring a dump to a directory other than the one it was dumped from. The working directory permissions and owner will then be set to those of the person doing the restore (typically **root**).

Various device names can access the **/dev/st0** device. Each name accesses a different minor device number that controls some aspect of how the tape drive is used. After you complete a dump when you use **/dev/st0**, the tape drive automatically rewinds the tape to the beginning. Use the nonrewinding SCSI tape device (**/dev/nst0**) to keep the tape from rewinding on completion. This feature allows you to back up multiple filesystems to one volume. Following is an example of backing up a system where the **/home**, **/usr**, and **/var** directories reside on different filesystems:

```
# dump -0uf /dev/nst0 /home
# dump -0uf /dev/nst0 /usr
# dump -0uf /dev/st0 /var
```

The preceding example uses the nonrewinding device for the first two dumps. If you use the rewinding device, the tape rewinds after each dump, and you are left with only the last dump on the tape.

You can use mt (magnetic tape), which is part of the **mt-st** package, to manipulate files on a multivolume dump tape. The following mt command positions the tape (**fsf 2** instructs mt to skip forward *past* two files, leaving the tape at the start of the third file). The restore command restores the **/var** filesystem from the previous example:

```
# mt -f /dev/st0 fsf 2
# restore rf /dev/st0
```

SCHEDULING TASKS

It is a good practice to schedule certain routine tasks to run automatically. For example, you may want to remove old core files once a week, summarize accounting data daily, and rotate system log files monthly.

crond AND crontab: SCHEDULE ROUTINE TASKS

Using crontab, you can submit a list of commands in a format that can be read and executed by **crond**. Working as Superuser, you can put commands in one of the **/etc/cron.*** directories to be run at intervals specified by the directory name, such as **cron.daily**.

When SELinux is set to use a targeted policy, it protects the **cron** daemon. You can disable this protection if necessary. For more information refer to "Setting the Targeted Policy with system-config-selinux" on page 416.

cron stops for no one; try anacron

tip The **crond** daemon assumes the system is always running. A similar utility, anacron, does not make that assumption and is well suited to portable and home computers that are frequently turned off. The anacron utility takes its instructions from the **/etc/anacrontab** file unless you specify otherwise. Refer to the anacron and anacrontab man pages for more information.

at: RUNS OCCASIONAL TASKS

Like the cron utility, at allows you to run a job sometime in the future. Unlike cron, at runs a job only once. For instance, you can schedule an at job that will reboot the system at 3 AM (when all users are probably logged off):

```
# at 3am
at> reboot
at> CONTROL-D <EOT>
job 1 at 2006-02-01 03:00
```

It is also possible to run an at job from within an at job. For instance, an at job might check for new patches every 18 days, something that would be more difficult with cron.

SYSTEM REPORTS

Many utilities report on one thing or another. The who, finger, ls, ps, and other utilities generate simple end-user reports. In some cases, these reports can help with system administration. This section describes utilities that generate more in-depth reports that can usually provide more assistance with system administration tasks. Linux has many other report utilities, including (from the **sysstat** package) sar (system activity report), iostat (input/output and CPU statistics), and mpstat (processor statistics); (from the **net-tools** package) netstat (network report); and (from the **nfs-utils** package) nfsstat (NFS statistics).

vmstat: REPORTS VIRTUAL MEMORY STATISTICS

The vmstat utility (**procps** package) generates virtual memory information along with (limited) disk and CPU activity data. The following example shows virtual memory statistics in 3-second intervals for seven iterations (from the arguments **3 7**). The first line covers the time since the system was last booted; the rest of the lines cover the period since the previous line.

```
$ vmstat 3 7
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 0  2      0 684328  33924 219916    0    0   430   105 1052   134  2  4 86  8
 0  2      0 654632  34160 248840    0    0  4897  7683 1142   237  0  5  0 95
 0  3      0 623528  34224 279080    0    0  5056  8237 1094   178  0  4  0 95
 0  2      0 603176  34576 298936    0    0  3416   141 1161   255  0  4  0 96
 0  2      0 575912  34792 325616    0    0  4516  7267 1147   231  0  4  0 96
 1  2      0 549032  35164 351464    0    0  4429    77 1120   210  0  4  0 96
 0  2      0 523432  35448 376376    0    0  4173  6577 1135   234  0  4  0 95
```

Table 16-2 lists the column heads displayed by vmstat.

Table 16-2 vmstat column heads

procs		**Process information**
	r	Number of waiting, runnable processes
	b	Number of blocked processes (in uninterruptable sleep)
memory		**Memory information in kilobytes**
	swpd	Used virtual memory
	free	Idle memory
	buff	Memory used as buffers
	cache	Memory used as cache

Table 16-2 vmstat column heads (continued)

swap		**System paging activity in kilobytes per second**
	si	Memory swapped in from disk
	so	Memory swapped out to disk
io		**System I/O activity in blocks per second**
	bi	Blocks received from a block device
	bo	Blocks sent to a block device
system		**Values are per second**
	in	Interrupts (including the clock)
	cs	Context switches
cpu		**Percentage of total CPU time spent in each of these states**
	us	User (nonkernel)
	sy	System (kernel)
	id	Idle
	wa	Waiting for I/O

top: LISTS PROCESSES USING THE MOST RESOURCES

The top utility is a useful supplement to ps. At its simplest, top displays system information at the top and the most CPU-intensive processes below the system information. The top utility updates itself periodically; type **q** to quit. Although you can use command-line options, the interactive commands are often more helpful. Refer to Table 16-3 (on the next page) and to the top man page for more information.

```
$ top
top - 21:30:26 up 18 min,  2 users,  load average: 0.95, 0.30, 0.14
Tasks:  63 total,   4 running,  58 sleeping,   1 stopped,   0 zombie
Cpu(s): 30.9% us, 22.9% sy, 0.0% ni,  0.0% id, 45.2% wa, 1.0% hi, 0.0%si
Mem:   1036820k total,  1032276k used,     4544k free,    40908k buffers
Swap:  2048276k total,        0k used,  2048276k free,   846744k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1285 root      25   0  9272 6892 1312 R 29.3  0.7   0:00.88 bzip2
 1276 root      18   0  3048  860 1372 R  3.7  0.1   0:05.25 cp
    7 root      15   0     0    0    0 S  0.7  0.0   0:00.27 pdflush
    6 root      15   0     0    0    0 S  0.3  0.0   0:00.11 pdflush
    8 root      15   0     0    0    0 S  0.3  0.0   0:00.06 kswapd0
  300 root      15   0     0    0    0 S  0.3  0.0   0:00.24 kjournald
 1064 mgs2      16   0  8144 2276 6808 S  0.3  0.2   0:00.69 sshd
 1224 root      16   0  4964 1360 3944 S  0.3  0.1   0:00.03 bash
 1275 mgs2      16   0  2840  936 1784 R  0.3  0.1   0:00.15 top
 1284 root      15   0  2736  668 1416 S  0.3  0.1   0:00.01 tar
    1 root      16   0  2624  520 1312 S  0.0  0.1   0:06.51 init
```

Table 16-3 top: interactive commands

Command	Function
F	Specify a sort field.
h or **?**	Displays a Help screen.
k	Prompts for a PID number and type of signal and sends the process that signal. Defaults to signal 15 (SIGTERM); specify 9 (SIGKILL) only when 15 does not work.
M	Sorts processes by memory usage.
O	Specify a sort field.
P	Sorts processes by CPU usage (default).
q	Quits.
s	Prompts for time between updates in seconds. Use 0 for continuous updates.
SPACE	Updates the display immediately.
T	Sorts tasks by time.
W	Writes a startup file named **~/.toprc** so that next time you start top, it uses the same parameters it is currently using.

parted: REPORTS ON AND PARTITIONS A HARD DISK

The parted (partition editor) utility reports on and manipulates hard disk partitions. The following example shows how to use parted from the command line (see "Running Commands from a Terminal Emulator/Shell" on page 118). It uses the **print** command to display information about the partitions on the **/dev/sda** drive:

```
# parted /dev/sda print
Disk geometry for /dev/sda: 0kB - 165GB
Disk label type: msdos
Number  Start   End     Size    Type      File system  Flags
1       32kB    1045MB  1045MB  primary   ext3         boot
2       1045MB  12GB    10GB    primary   ext3
3       12GB    22GB    10GB    primary   ext3
4       22GB    165GB   143GB   extended
5       22GB    23GB    1045MB  logical   linux-swap
6       23GB    41GB    18GB    logical   ext3
7       41GB    82GB    41GB    logical   ext3
Information: Don't forget to update /etc/fstab, if necessary.
```

Figure 16-3 graphically depicts the partitions shown in this example. The first line that parted displays specifies the device being reported on (**/dev/sda**) and its size (165 gigabytes). The **print** command displays the following columns:

- **Number**—The minor device number (page 484) of the device holding the partition. This number is the same as the last number in the device name. In the example, **5** corresponds to **/dev/sda5**.

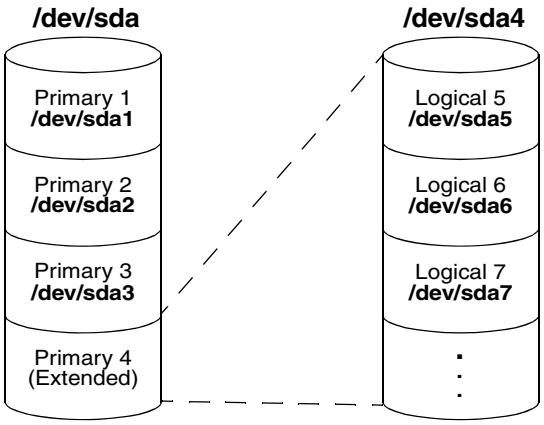

Figure 16-3 The primary and extended partitions from the example

- **Start**—The location on the disk where the partition starts. The parted utility specifies a location on the disk as the distance (in bytes) from the beginning of the disk. Thus partition 3 starts 12 gigabytes from the beginning of the disk.
- **End**—The location on the disk where the partition stops. Although partition 2 ends 12 gigabytes from the beginning of the disk and partition 3 starts at the same location, parted takes care that the partitions do not overlap at this single byte.
- **Size**—The size of the partition in kilobytes (kB), megabytes (MB), or gigabytes (GB).
- **Type**—The partition type: primary, extended, or logical. See Figure 16-3 and page 31 for information on partition types.
- **File system**—The filesystem type: **ext2**, **ext3**, **fat32**, **linux-swap**, and so on. See Table 12-1 on page 485 for a list of filesystem types.
- **Flags**—The flags that are turned on for the partition, including **boot**, **raid**, and **lvm**. In the example, partition 1 is bootable.

In the preceding example, partition 4 defines an extended partition that includes 143 gigabytes of the 165-gigabyte disk (Figure 16-3). You cannot make changes to an extended partition without affecting all logical partitions within it.

In addition to reporting on the layout and size of a hard disk, you can use parted interactively to modify the disk layout. Be *extremely* careful when using parted in this manner, and always back up the system before you work with this utility. Changing the partition information (the *partition table*) on a disk can destroy the information on the disk. Read the parted info page before you attempt to modify a partition table.

parted can destroy everything

caution Be as careful with parted as you would be with a utility that formats a hard disk. Changes you make with parted can easily result in the loss of large amounts of data. If you are using parted and have any question about what you are doing, quit with a **q** command before making any changes. Once you give parted a command, it immediately makes the change you requested.

To partition a disk, give the command **parted** followed by the name of the device you want to work with. In the following example, after starting parted, the user gives a **help** (or just **h**) command, which displays a list of parted commands:

```
# parted /dev/hdb
GNU Parted 1.8.6
Using /dev/hdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) help
  check NUMBER                  do a simple check on the file system
  cp [FROM-DEVICE] FROM-NUMBER TO-NUMBER   copy file system to another partition
  help [COMMAND]                prints general help, or help on COMMAND
  mklabel LABEL-TYPE            create a new disklabel (partition table)
  mkfs NUMBER FS-TYPE           make a FS-TYPE file system on partititon NUMBER
  mkpart PART-TYPE [FS-TYPE] START END     make a partition
  mkpartfs PART-TYPE FS-TYPE START END     make a partition with a file system
  move NUMBER START END         move partition NUMBER
  name NUMBER NAME              name partition NUMBER as NAME
  print [NUMBER]                display the partition table, or a partition
  quit                          exit program
  rescue START END              rescue a lost partition near START and END
  resize NUMBER START END       resize partition NUMBER and its file system
  rm NUMBER                     delete partition NUMBER
  select DEVICE                 choose the device to edit
  set NUMBER FLAG STATE         change a flag on partition NUMBER
  toggle [NUMBER [FLAG]]        toggle the state of FLAG on partition NUMBER
  unit UNIT                     set the default unit to UNIT
  version                       displays the version of GNU Parted and copyright info
(parted)
```

In response to the (**parted**) prompt, you can give the command **help** followed by the name of the command you want more information about. When you give a **print** (or just **p**) command, parted displays current partition information, just as a **print** command on the command line does.

The parted utility will not allow you to set up overlapping partitions (except for logical partitions that overlap the extended partition that contains them). Similarly it will not allow you to create a partition that starts at the very beginning of the disk (cylinder 0). Both of these situations can cause loss of data.

Following are guidelines to remember when defining a partition table for a disk. For more information refer to "Setting Up the Hard Disk" on page 30.

- Do not delete or modify the partition that defines the extended partition unless you are willing to lose all data on all logical partitions within the extended partition.

- If you put **/boot** on a separate partition, it is a good idea to put it at the beginning of the drive (partition 1) so there is no issue of Linux having to boot from a partition located too far into the drive. When you can afford the disk space, it is desirable to put each major filesystem on a separate partition. Many people choose to combine **/** (root), **/var**, and **/usr** into a single partition, which generally results in less wasted space but can, on rare occasions, cause problems.
- Although parted can create some types of filesystems, it is typically easiest to use this utility to create partitions and then use mkfs and mkswap to create filesystems on the partitions.

The following sequence of commands defines a 300-megabyte, bootable, Linux partition as partition 1 on a clean disk:

```
# parted /dev/hdb
...
Using /dev/hdb
(parted) mkpart                                  (create new partition)
Partition type?  primary/extended? primary       (select primary partition)
File system type?  [ext2]?                       (default to an ext2 filesystem)
Start? 1                                         (start at the beginning of the disk)
End? 300m                                        (specify a 300-megabyte partition)
(parted) help set                                (use help to check the syntax of the set command)
  set NUMBER FLAG STATE         change a flag on partition NUMBER

        NUMBER is the partition number used by Linux.  On msdos disk labels, the primary
        partitions number from 1 to 4, logical partitions from 5 onwards.
        FLAG is one of: boot, root, swap, hidden, raid, lvm, lba, hp-service, palo,
        prep, msftres
        STATE is one of: on, off
(parted) set 1 boot on                           (turn on the boot flag on partition 1)
(parted) print                                   (verify that the partition is correct)
Disk geometry for /dev/hdb: 0kB - 250GB
Disk label type: msdos
Number  Start   End     Size    Type      File system  Flags
1       1kB     300MB   300MB   primary   ext2         boot
(parted) quit
Information: Don't forget to update /etc/fstab, if necessary.
```

When you specify a size within parted, you can use a suffix of **k** (kilobytes), **m** (megabytes), or **g** (gigabytes). After creating a partition, give a **print** command to see where the partition ends. Perform this task before you define the next contiguous partition to make sure you do not waste space. After setting up all the partitions, exit from parted with a **quit** command.

Next make a filesystem (mkfs; page 439) on each partition that is to hold a filesystem (not swap). Make all partitions, except swap and **/boot**, of type **ext3**, unless you have a reason to do otherwise. Make the **/boot** partition of type **ext2**. Use mkswap (page 479) to set up a swap area on a partition. You can use e2label (page 439) to label partitions.

KEEPING USERS INFORMED

One of your primary responsibilities as a system administrator is communicating with system users. You need to make announcements, such as when the system will be down for maintenance, when a class on some new software will be held, and how users can access the new system printer. You can even start to fill the role of a small local newspaper, letting users know about new employees, RIFs, births, the company picnic, and so on.

Different communications have different priorities. For example, information about the company picnic in two months is not as time sensitive as the fact that you are bringing the system down in 5 minutes. To meet these differing needs, Linux provides different ways of communicating. The most common methods are described and contrasted in the following list. All of these methods are generally available to everyone, except for the message of the day, which is typically reserved for Superuser.

write Use the write utility (page 172) to communicate with a user who is logged in on the local system. You might use it, for example, to ask a user to stop running a program that is bogging down the system; the user might reply that he will be done in 3 minutes. Users can also use write to ask the system administrator to mount a tape or restore a file. GNOME opens a new window when it receives a message.

wall The wall (write all) utility effectively communicates immediately with all users who are logged in. It works similarly to write, except that users cannot use wall to write back to only you. Use wall when you are about to bring the system down or are in another crisis situation. Users who are not logged in will not get the message.

Use wall while you are Superuser *only* in a crisis situation; it interrupts anything anyone is doing.

Email Email is useful for communicating less urgent information to one or more systems and/or remote users. When you send mail, you have to be willing to wait for each user to read it. The email utilities are useful for reminding users that they are forgetting to log out, their bills are past due, or they are using too much disk space.

Users can easily make permanent records of messages they receive via email, as opposed to messages received via write, so they can keep track of important details. It would be appropriate to use email to inform users about a new, complex procedure, so each user could keep a copy of the information for reference.

Message of the day Users see the message of the day each time they log in in a textual environment. You can edit the **/etc/motd** file to change this message as necessary. The message of the day can alert users to upcoming periodic maintenance, new system features, or a change in procedures.