LPI exam 102 prep, Topic 111: Administrative tasks

Junior Level Administration (LPIC-1) topic 111

Skill Level: Intermediate

Ian Shields (ishields@us.ibm.com)
Senior Programmer
IBM

10 Jul 2007

In this tutorial, Ian Shields continues preparing you to take the Linux Professional Institute® Junior Level Administration (LPIC-1) Exam 102. In this sixth in a series of nine tutorials, Ian introduces you to administrative tasks. By the end of this tutorial, you will know how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

Section 1. Before you start

Learn what these tutorials can teach you and how you can get the most from them.

About this series

The Linux Professional Institute (LPI) certifies Linux system administrators at three levels: *junior level* (also called "certification level 1"), *intermediate level* (also called "certification level 2"), and *senior level* (also called "certification level 3"). To attain certification level 1, you must pass exams 101 and 102; to attain certification level 2, you must pass exams 201 and 202. To attain certification level 3, you must have an active intermediate level certification and pass exam 301 ("core"). You may also pass additional specialty exams at the senior level.

developerWorks offers tutorials to help you prepare for the four junior and intermediate certification exams. Each exam covers several topics, and each topic has a corresponding self-study tutorial on developerWorks. For LPI exam 102, the nine topics and corresponding developerWorks tutorials are:

Table 1. LPI exam 102: Tutorials and topics

LPI exam 102 topic	developerWorks tutorial	Tutorial summary
Topic 105	LPI exam 102 prep: Kernel	Learn how to install and maintain Linux kernels and kernel modules.
Topic 106	LPI exam 102 prep: Boot, initialization, shutdown, and runlevels	Learn how to boot a system, set kernel parameters, and shut down or reboot a system.
Topic 107	LPI exam 102 prep: Printing	Learn how to manage printers, print queues and user print jobs on a Linux system.
Topic 108	LPI exam 102 prep: Documentation	Learn how to use and manage local documentation, find documentation on the Internet and use automated logon messages to notify users of system events.
Topic 109	LPI exam 102 prep: Shells, scripting, programming, and compiling	Learn how to customize shell environments to meet user needs, write Bash functions for frequently used sequences of commands, write simple new scripts, using shell syntax for looping and testing, and customize existing scripts.
Topic 111	LPI exam 102 prep: Administrative tasks	(This tutorial.) Learn how to manage user and group accounts and tune user and system environments, configure and use system log files, automate system administration tasks by scheduling jobs to run at another time, back up your system, and maintain system time. See the detailed objectives below.
Topic 112	LPI exam 102 prep: Networking fundamentals	Coming soon.
Topic 113	LPI exam 102 prep: Networking services	Coming soon.
Topic 114	LPI exam 102 prep: Security	Coming soon.

To pass exams 101 and 102 (and attain certification level 1), you should be able to:

- Work at the Linux command line
- Perform easy maintenance tasks: help out users, add users to a larger system, back up and restore, and shut down and reboot
- Install and configure a workstation (including X) and connect it to a LAN, or connect a stand-alone PC via modem to the Internet

To continue preparing for certification level 1, see the developerWorks tutorials for LPI exams 101 and 102, as well as the entire set of developerWorks LPI tutorials.

The Linux Professional Institute does not endorse any third-party exam preparation material or techniques in particular. For details, please contact info@lpi.org.

About this tutorial

Welcome to "Administrative tasks," the sixth of nine tutorials designed to prepare you for LPI exam 102. In this tutorial, you learn how to manage users and groups, set user profiles and environments, use log files, schedule jobs, back up your data, and maintain the system time.

This tutorial is organized according to the LPI objectives for this topic. Very roughly, expect more questions on the exam for objectives with higher weight.

Table 2. Administrative tasks: Exam objectives covered in this tutorial

LPI exam objective	Objective weight	Objective summary
1.111.1 User and group accounts	Weight 4	Add, remove, suspend, and change user accounts. Manage user and group information in password and group databases, including shadow databases. Create and manage special purpose and limited accounts.
1.111.2 Tune user and system environments	Weight 3	Modify global and user profiles. Set environment variables and maintain skeleton directories for new user accounts. Set command search paths.
1.111.3 Configure and use system log files to meet administrative and security needs	Weight 3	Configure and manage system logs, including the type and level of logged information. Scan and monitor log files for

		notable activity and track down noted problems. Rotate and archive log files.
1.111.4 Automate system administration tasks by scheduling jobs to run in the future	Weight 4	Use the `cron` or `anacron` commands to run jobs at regular intervals, and use the `at` command to run jobs at a specific time.
1.111.5 Maintain an effective data backup strategy	Weight 3	Plan a backup strategy and back up filesystems automatically to various media.
1.111.6 Maintain system time	Weight 4	Maintain the system time and time zone, and synchronize the clock via NTP. Set the BIOS clock to the correct time in UTC, and configure NTP, including correcting for clock drift.

Prerequisites

To get the most from this tutorial, you should have a basic knowledge of Linux and a working Linux system on which to practice the commands covered in this tutorial.

This tutorial builds on content covered in previous tutorials in this LPI series, so you may want to first review the tutorials for exam 101. In particular, you should be thoroughly familiar with the material from the "LPI exam 101 prep (topic 104) Devices, Linux filesystems, and the Filesystem Hierarchy Standard" tutorial, which covers basic concepts of users, groups, and file permissions.

Different versions of a program may format output differently, so your results may not look exactly like the listings and figures in this tutorial.

Section 2. User and group accounts

This section covers material for topic 1.111.1 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Add, modify, and remove users and groups
- Suspend and change user accounts
- Manage user and group information in the password databases and group databases
- Use the correct tools to manage shadow password databases and group databases
- Create and manage limited and special-purpose accounts

As you learned in the "LPI exam 101 prep (topic 104) Devices, Linux filesystems, and the Filesystem Hierarchy Standard" tutorial, Linux is a multi-user system where each user belongs to one *primary* group and possibly to additional groups. Ownership of files in Linux is closely related to user ids and groups. Recall that you can log in as one user and become another user using the `su` or `sudo -s`commands, and that you can use the `whoami` command to check your current effective id and the `groups` command to find out what groups you belong to. In this section, you learn how to create, delete, and manage users and groups. You also learn about the files in /etc, where user and group information is stored.

Add and remove users and groups

You add a user to a Linux system using the `useradd` command, and you delete a user using the `userdel` command. Similarly, you add or delete groups using the `groupadd` and `groupdel` commands.

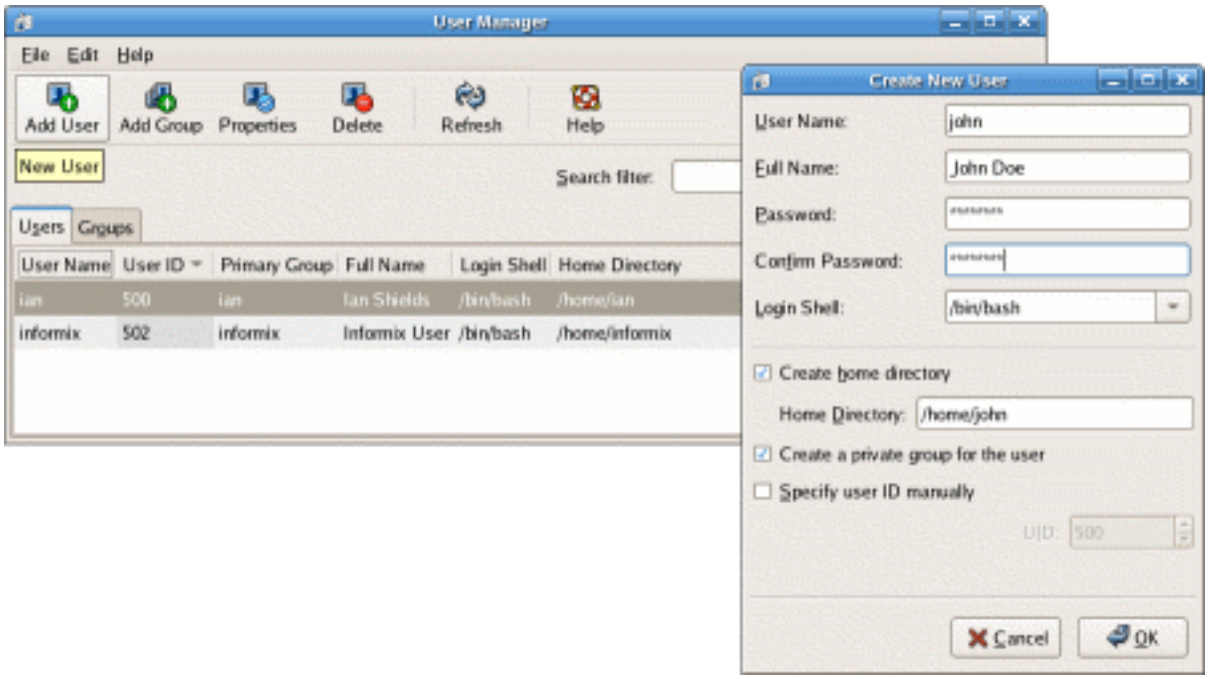
Adding a user or group

Modern Linux desktops usually have graphical interfaces for user and group administration. The graphical interface is usually accessed through menu options for system administration. These interfaces do vary considerably, so the one on your system may not look much like the example here, but the underlying concepts and commands remain similar.

Let's start by adding a user to a Fedora Core 5 system graphically, and then examine the underlying commands. In the case of Fedora Core 5 with GNOME desktop, use **System > Administration > Users and Groups**, then click the **Add User** button.

Figure 1 depicts the User Manager panel with the Create New User panel showing basic information for a new user named 'john'. The full name of the user, John Doe, and a password have been entered. The panel provides a default login shell of /bin/bash. On Fedora systems, the default is to create a new group with the same name as the user, 'john' in this case, and a home directory of /home/john.

Figure 1. Adding a user



Listing 1 shows the use of the `id` command to display basic information about the new user. As you can see, john has user number 503 and a matching group, john, with group number 503. This is the only group of which john is a member.

Listing 1. Displaying user id information

```
[root@pinguino ~]# id john
uid=503(john) gid=503(john) groups=503(john)
```

To accomplish the same task from the command line, you use the `groupadd` and `useradd` commands to create the group and user, then use the `passwd` command to set the password for the newly created user. All of these commands require root authority. The basic use of these commands to add another user, jane, is illustrated in Listing 2.

Listing 2. Adding user jane

```
[root@pinguino ~]# groupadd jane
[root@pinguino ~]# useradd -c "Jane Doe" -g jane -m jane
[root@pinguino ~]# passwd jane
Changing password for user jane.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# id jane
uid=504(jane) gid=504(jane) groups=504(jane)
[root@pinguino ~]# ls -ld /home/jane
drwx------ 3 jane jane 4096 Jun 25 18:22 /home/jane
```

In these two examples, both the user id and the group id have values greater than 500. Be aware that some newer systems start user ids at 1000 rather than 500. These values normally signify ordinary users, while values below 500 (or 1000 if the system starts ordinary users at 1000) are reserved for *system users*. System users are covered later in this section. The actual cutoff points are set in /etc/login.defs as `UID_MIN` and `GID_MIN`.

In Listing 2 above, the `groupadd` command has a single parameter, `jane`, the name of the group to be added. Group names must begin with a lower case letter or an underscore, and usually contain only these along with hyphens or dashes. Options you may specify are shown in Table 3.

Table 3. Options for groupadd

Option	Purpose
-f	Exit with success status if the group already exists. This is handy for scripting when you do not need to check if a group exists before attempting to create it.
-g	Specifies the group id manually. The default is to use the smallest value that is at least `GID_MIN` and also greater than the id of any existing group. Group ids are normally unique and must be non-negative
-o	Permits a group to have a non-unique id.
-K	Can be used to override defaults from /etc/login.defs.

In Listing 2 above, the `useradd` command has a single parameter, `jane`, the name of the user to be added, along with the `-c`, `-g`, and `-m` options. Common options for the `useradd` command are shown in Table 4.

Table 4. Options for useradd

Option	Purpose
-b --base-dir	The default base directory in which user home directories are created. This is usually /home, and the user's home directory is /home/$USER.
-c --comment	A text string describing the id, such as the user's full name.
-d --home	Provides a specific directory name for the home directory.
-e	The date on which the account will

--expiredate	expire or be disabled in the form YYYY-MM_DD.
-g --gid	The name or number of the initial login group for the user. The group must exist, which is why group jane was created before user jane in Listing 2.
-G --groups	A comma-separated list of additional groups to which the user belongs.
-K	Can be used to override defaults from /etc/login.defs.
-m --create-home	Create the user's home directory if it does not exist. Copy the skeleton files and any directories from /etc/skel to the home directory.
-o --non-unique	Permits a user to have a non-unique id.
-p --password	The encrypted password. If a password is not specified, the default is to disable the account. You will usually use the `passwd` command in a subsequent step rather than generating an encrypted password and specifying it on the `useradd` command.
-s --shell	The name of the user's login shell if different from the default login shell.
-u --uid	The non-negative numerical userid, which must be unique if `-o` is not specified. The default is to use the smallest value that is at least `UID_MIN` and also greater than the id of any existing user.

Notes:

1. Some systems, including Fedora and Red Hat distributions, have extensions to the user-creation commands. For example, the default Fedora and Red Hat behavior is to create a new group for a user, and the `-n` option can be used on the `useradd` command to disable this function. Be aware of such possible system differences and refer to the man pages on your system when in doubt.

2. On SUSE systems, use YaST or YaST2 to access graphical user and group administration interfaces.

3. Graphical interfaces may perform additional tasks such as creating the user's mail file in /var/spool/mail.

Deleting a user or group

Deleting a user or group is much simpler than adding one, because there are fewer options. In fact, the `groupdel` command to delete a group requires only the group name; it has no options. You cannot delete any group that is the primary group of a user. If you use a graphical interface for deleting users and groups, the functions are very similar to the commands shown here.

Use the `userdel` command to delete a user. The `-r`, or `--remove` option requests removal of the user's home directory and anything it contains, along with the user's mail spool. When you delete a user, a group with the same name as the user will also be deleted if USERGROUPS_ENAB is set to yes in /etc/login.defs, but this will be done only if the group is not the primary group of another user.

In Listing 3 you see an example of deleting groups when multiple users share the same primary group. Here, another user, jane2, has previously been added to the system with the same group as jane.

Listing 3. Deleting users and groups

```
root@pinguino:~# groupdel jane
groupdel: cannot remove user's primary group.
root@pinguino:~# userdel -r jane
userdel: Cannot remove group jane which is a primary group for another
user.
root@pinguino:~# userdel -r jane2
root@pinguino:~# groupdel jane
```

Notes:

1. There is a `userdel` option, `-f` or `--force`, which can be used to delete users and their group. This option is dangerous, so you should use it only as a last resort. Read the man page carefully before you do.

2. Be aware that if you delete a user or group, and there are files that belong to that user or group on your filesystem, then the files are not automatically deleted or assigned to another user or group.

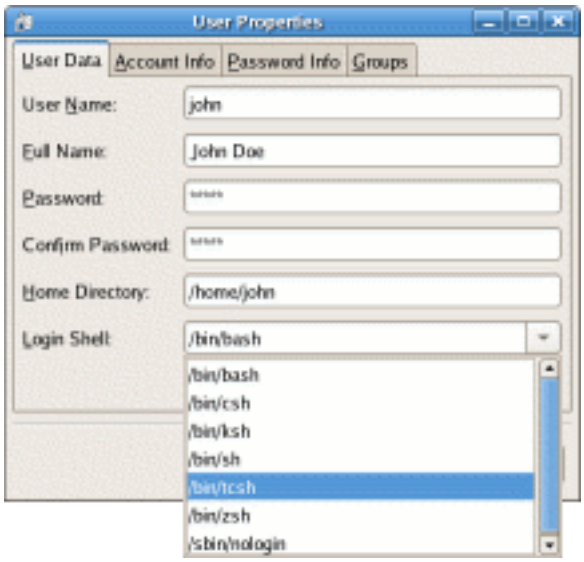
Suspend and change accounts

Now that you can create or delete a user id or a group, you may also find a need to modify one.

Modifying user accounts

Suppose user john wishes to have the tcsh shell as his default. From a graphical interface you will usually find a way to either edit a user (or group), or to examine the properties of the object. Figure 2 shows the properties dialog for the user john that we created earlier on a Fedora Core 5 system.

Figure 2. Modifying a user account



From the command line, you can use the `usermod` command to modify a user account. You can use most of the options that you use with `useradd`, except that you cannot create or populate a new home directory for the user. If you need to change the name of the user, specify the `-l` or `--login` option with the new name. You will probably want to rename the home directory to match the user id. You may also need to rename other items such as mail spool files. Finally, if the login shell is changed, some of the associated profile files may need to be altered. Listing 4 shows an example of the things you might need to do to change user john to john2 with /bin/tcsh as the default shell and renamed home directory /home/john2.

Listing 4. Modifying a user

```
[root@pinguino ~]# usermod -l john2 -s /bin/tcsh -d /home/john2 john
[root@pinguino ~]# ls -d ~john2
ls: /home/john2: No such file or directory
[root@pinguino ~]# mv /home/john /home/john2
[root@pinguino ~]# ls -d ~john2
/home/john2
```

Notes:

1. If you need to modify a user's additional groups, you must specify the complete list of additional groups. There is no command to simply add or delete a single group for a user.

2. There are restrictions on changing the name or id of a user who is logged in or who has running processes. Check the man pages for details.

3. If you change a user number, you may want to change files and directories owned by that user to match the new number.

Modifying groups

Not surprisingly, the `groupmod` command is used to modify group information. You can change the group number with the `-g` option, and the name with the `-n` option.

Listing 5. Renaming a group

```
[root@pinguino ~]# ls -ld ~john2
drwx------ 3 john2 john 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# groupmod -n john2 john
[root@pinguino ~]# ls -ld ~john2
drwx------ 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

Notice in Listing 5 that the group name for the home directory of john2 magically changed when we used `groupmod` to change the group name. Are you surprised? Because groups are represented in the filesystem inodes by their number rather than by their name, this is not surprising. However, if you change a group's number, you should update any users for which that group is the primary group, and you may also want to update the files and directories belonging to that group to match the new number (in the same way as noted above for changing a user number). Listing 6 shows how to change the group number for john2 to 505, update the user account, and make appropriate changes to all the affected files in the /home filesystem. You probably want renumbering users and groups if at all possible.

Listing 6. Renumbering a group

```
[root@pinguino ~]# groupmod -g 505 john2
[root@pinguino ~]# ls -ld ~john2
drwx------ 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# id john2
uid=503(john2) gid=503 groups=503
[root@pinguino ~]# usermod -g john2 john2
[root@pinguino ~]# id john2
uid=503(john2) gid=505(john2) groups=505(john2)
[root@pinguino ~]# ls -ld ~john2
drwx------ 3 john2 503 4096 Jun 26 18:29 /home/john2
[root@pinguino ~]# find /home -gid 503 -exec chgrp john2 {} \;
[root@pinguino ~]# ls -ld ~john2
drwx------ 3 john2 john2 4096 Jun 26 18:29 /home/john2
```

User and group passwords

You have already seen the `passwd` command, which is used to change a user password. The password is (or should be) unique to the user and may be changed by user. The root user may change any user's password as we have already seen.

Groups may also have passwords, and the `gpasswd` command is used to set them. Having a group password allows users to join a group temporarily with the `newgrp` command, if they know the group password. Of course, having multiple people knowing a password is somewhat problematic, so you will have to weigh the advantages of adding a user to a group using `usermod`, versus the security issue of having too many people knowing the group password.

Suspending or locking accounts

If you need to prevent a user from logging in, you can *suspend* or *lock* the account using the `-L` option of the `usermod` command. To *unlock* the account, use the `-U` option. Listing 7 shows how to lock account john2 and what happens if john2 attempts to log in to the system. Note that when the john2 account is unlocked, the same password is restored.

Listing 7. Locking an account

```
[root@pinguino ~]# usermod -L john2
[root@pinguino ~]# ssh john2@pinguino
john2@pinguino's password:
Permission denied, please try again.
```

You may have noticed back in Figure 2 that there were several tabs on the dialog box with additional user properties. We briefly mentioned the use of the `passwd` command for setting user passwords, but both it and the `usermod` command can perform many tasks related to user accounts, as can another command, the `chage` command. Some of these options are shown in Table 5. Refer to the appropriate man pages for more details on these and other options.

Table 5. Commands and options for changing user accounts

Option for command			Purpose
Usermod	**Passwd**	**Chage**	
-L	-l	N/A	Lock or suspend the account.
-U	-u	N/A	Unlock the account.
N/A	-d	N/A	Disable the account by setting it passwordless.

-e	-f	-E	Set the expiration date for an account.
N/A	-n	-m	The minimum password lifetime in days.
N/A	-x	-M	The maximum password lifetime in days.
N/A	-w	-W	The number of days of warning before a password must be changed.
-f	-i	-I	The number of days after a password expires until the account is disabled.
N/A	-S	-l	Output a short message about the current account status.

Manage user and group databases

The primary repositories for user and group information are four files in /etc.

/etc/passwd
is the *password* file containing basic information about users

/etc/shadow
is the *shadow password* file containing encrypted passwords

/etc/group

is the *group* file containing basic information about groups and which users belong to them

/etc/gshadow
is the *shadow group* file containing encrypted group passwords

These files are updated by the commands you have already seen in this tutorial and you will meet some more commands for working with them after we discuss the files themselves. All of these files are plain text files. In general, you should not edit them directly. Use the tools provided for updating them so they are properly locked and kept synchronized.

You will note that the passwd and group files are both *shadowed*. This is for security reasons. The passwd and group files themselves must be world readable, but the encrypted passwords should not be world readable. Therefore, the shadow files contain the encrypted passwords, and these files are only readable by root. The necessary authentication access is provided by an suid program that has root authority, but can be run by anyone. Make sure that your system has the permissions set appropriately. Listing 8 shows an example.

Listing 8. User and group database permissions

```
[ian@pinguino ~]$ ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
-rw-r--r-- 1 root root  701 Jun 26 19:04 /etc/group
-r-------- 1 root root  580 Jun 26 19:04 /etc/gshadow
-rw-r--r-- 1 root root 1939 Jun 26 19:43 /etc/passwd
-r-------- 1 root root 1324 Jun 26 19:50 /etc/shadow
```

Note: Although it is still technically possible to run without shadowed password and group files, this is almost never done and is not recommended.

The /etc/passwd file

The /etc/passwd file contains one line for each user in the system. Some example lines are shown in Listing 9.

Listing 9. /etc/password entries

```
root:x:0:0:root:/root:/bin/bash
jane:x:504:504:Jane Doe:/home/jane:/bin/bash
john2:x:503:505:John Doe:/home/john2:/bin/tcsh
```

Each line contains seven fields separated by colons (:), as shown in Table 6.

Table 6. Fields in /etc/passwd

Field	Purpose
Username	The name used to log in to the system.

	For example, john2.
Password	The encrypted password. When using shadow passwords, it contains a single x character.
User id (UID)	The number used to represent this user name in the system. For example, 503 for user john2.
Group id (GID)	The number used to represent this user's primary group in the system. For example, 505 for user john2.
Comment (GECOS)	An optional field used to describe the user. For example, "John Doe". The field may contain multiple comma-separated entries. It is also used by programs such as `finger`. The GECOS name is historic. See details in `man 5 passwd`.
Home	The absolute path the user's home directory. For example, /home/john2
Shell	The program automatically launched when a user logs in to the system. This is usually an interactive shell such as /bin/bash or /bin/tcsh, but may be any program, not necessarily an interactive shell.

The /etc/group file

The /etc/group file contains one line for each group in the system. Some example lines are shown in Listing 10.

Listing 10. /etc/group entries

```
root:x:0:root
jane:x:504:john2
john2:x:505:
```

Each line contains four fields separated by colons (:), as shown in Table 7.

Table 7. Fields in /etc/group

Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The encrypted password. When using shadow group passwords, it contains a single x character.

Group id (GID)	The number used to represent this group in the system. For example, 505 for group john2.
Members	A comma-separated list of group members, excepting those members for whom this is the primary group.

Shadow files

The file /etc/shadow should only be readable by root. It contains encrypted passwords, along with password and account expiration information. See the man page (`man 5 shadow`) for information on the field layout. Passwords may be encrypted using DES, but are more usually encrypted using MD5. The DES algorithm uses the low order 7 bits of the first 8 characters of the user password as a 56-bit key, while the MD5 algorithm uses the whole password. In either case, passwords are *salted* so that two otherwise identical passwords do not generate the same encrypted value. Listing 11 shows how to set identical passwords for users jane and john2, and then shows the resulting encoded MD5 passwords in /etc/shadow.

Listing 11. Passwords in /etc/shadow

```
[root@pinguino ~]# echo lpic1111 |passwd jane --stdin
Changing password for user jane.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# echo lpic1111 |passwd john2 --stdin
Changing password for user john2.
passwd: all authentication tokens updated successfully.
[root@pinguino ~]# grep "^j" /etc/shadow
jane:$1$eG0/KGQY$ZJl.ltYtVw0sv.C5OrqUu/:13691:0:99999:7:::
john2:$1$grkxo6ie$J2muvoTpwo3dZAYYTDYNu.:13691:0:180:7:29::
```

The leading 1 indicates an MD5 password, and the salt is a variable length field of up to 8 characters ending with the next $ sign. The encrypted password is the remaining string of 22 characters.

Tools for users and groups

You have already seen several commands that manipulate the account and group files and their shadows. Here you learn about:

- Group administrators
- Editing commands for password and group files
- Conversion programs

Group administrators

In some circumstances you may want users other than root to be able to administer one or more groups by adding or removing group members. Listing 12 shows how root can add user jane as an administrator of group john2, and then jane, in turn, can add user ian as a member.

Listing 12. Adding group administrators and members

```
[root@pinguino ~]# gpasswd -A jane john2
[root@pinguino ~]# su - jane
[jane@pinguino ~]$ gpasswd -a ian john2
Adding user ian to group john2
[jane@pinguino ~]$ id ian;id jane
uid=500(ian) gid=500(ian) groups=500(ian),505(john2)
uid=504(jane) gid=504(jane) groups=504(jane)
```

You may be surprised to note that, although jane is an administrator of group john2, she is not a member of it. An examination of the structure of /etc/gshadow shows why. The /etc/gshadow file contains four fields for each entry as shown in Table 8. Note that the third field is a comma-separated list of administrators for the group.

Table 8. Fields in /etc/gshadow

Field	Purpose
Groupname	The name of this group. For example, john2.
Password	The field used to contain the encrypted password if the group has a password. If there is no password, you may see 'x', '!' or '!!' here.
Admins	A comma-separated list of group administrators.
Members	A comma-separated list of group members.

As you can see, the administrator list and the member list are two distinct fields. The `-A` option of `gpasswd` allows the root user to add administrators to a group, while the `-M` option allows root to add members. The `-a` (note lower case) option allows an administrator to add a member, while the `-d` option allows an administrator to remove a member. Additional options allow a group password to be removed. See the man pages for details.

Editing commands for password and group files

Although not listed in the LPI objectives, you should also be aware of the `vipw` command for safely editing /etc/passwd and `vigr` for safely editing /etc/group. The

commands will lock the necessary files while you make changes using the `vi` editor. If you make changes to /etc/passwd, then `vipw` will prompt you to see if you also need to update /etc/shadow. Similarly, if you update /etc/group using `vigr`, you will be prompted to update /etc/gshadow. If you need to remove group administrators, you may need to use `vigr`, as `gpasswd` only allows addition of administrators.

Conversion programs

Four other related commands are also not listed in the LPI objectives. They are `pwconv`, `pwunconv`, `grpconv`, and `grpunconv`. They are used for converting between shadowed and non-shadowed password and group files. You may never need these, but be aware of their existence. See the man pages for details.

Limited and special-purpose accounts

By convention, system users usually have an id of less than 100, with root having id 0. Normal users start automatic numbering from the `UID_MIN` value set in /etc/login.defs, with this value commonly being set at 500 or 1000.

Besides regular user accounts and the root account on your system, you will usually have several special-purpose accounts, for daemons such as FTP, SSH, mail, news, and so on. Listing 13 shows some entries from /etc/passwd for these.

Listing 13. Limited and special-purpose accounts

```
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
```

Such accounts frequently control files but should not be accessed by normal login. Therefore, they usually have a login shell specified as /sbin/nologin, or /bin/false so that login attempts will fail.

Section 3. Environment tuning

This section covers material for topic 1.111.2 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to tune the user environment, including these tasks:

- Set and unset environment variables
- Maintain skeleton directories for new user accounts
- Set command search paths

Set and unset environment variables

When you create a new user, you usually initialize many variable according to your local needs. These are usually set in the profiles that you provide for new users, such as .bash_profile and .bashrc, or in the system-wide profiles /etc/profile and /etc/bashrc. Listing 14 shows an example of how the PS1 system prompt is set in /etc/profile on an Ubuntu 7.04 system. The first `if` statement checks whether the PS1 variable is set, indicating an interactive shell, since a non-interactive shell doesn't need a prompt. The second `if` statement checks whether the BASH environment variable is set. If so, it sets a complex prompt and sources (note the dot) /etc/bash.bashrc. If the BASH variable is not set, then a check is made for root (id=0), and the prompt is set to # or $ accordingly.

Listing 14. Setting environment variables

```
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
        . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
```

The tutorial LPI exam 102 prep: Shells, scripting, programming, and compiling has detailed information about the commands used for setting and unsetting environment variables, as well as information about how and when the various profiles are used.

When customizing environments for users, be aware of two major points:

1. /etc/profile is read only at login time, and so it is not executed when each new shell is created.

2. Functions and aliases are not inherited by new shells. Therefore, you will usually set these and your environment variables in /etc/bashrc, or in the user's own profiles.

In addition to the system profiles, /etc/profile and /etc/bashrc, the Linux Standard Base (LSB) specifies that additional scripts may be placed in the directory /etc/profile.d. These scripts are sourced when an interactive login shell is created. They provide a convenient way of separating customization for different programs. Listing 15 shows an example.

Listing 15. /etc/profile.d/vim.sh on Fedora 7

```
[if [ -n "$BASH_VERSION" -o -n "$KSH_VERSION" -o -n "$ZSH_VERSION" ];
then
  [ -x //usr/bin/id ] || return
  [ `//usr/bin/id -u` -le 100 ] && return
  # for bash and zsh, only if no alias is already set
  alias vi >/dev/null 2>&1 || alias vi=vim
fi
```

Remember that you should usually `export` any variables that you set in a profile; otherwise, they will not be available to commands that run in a new shell.

Maintain skeleton directories for new users

You learned in the section Add and remove users and groups that you can create or populate a new home directory for the user. The source for this new directory is the subtree rooted at /etc/skel. Listing 16 shows the files in this subtree for a Fedora 7 system. Note that most files start with a period (dot), so you need the `-a` option to list them. The `-R` options lists subdirectories recursively, and the `-L` option follows any symbolic links.

Listing 16. /etc/skel on Fedora 7

```
[ian@lyrebird ~]$ ls -aRL /etc/skel
/etc/skel:
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .xemacs

/etc/skel/.xemacs:
.  ..  init.el
```

In addition to .bash_logout, .bash_profile, and .bashrc, which you might expect for the Bash shell, note that this example includes profile information for the emacs and xemacs editors. See the tutorial LPI exam 102 prep: Shells, scripting, programming, and compiling if you need to review the functions of the various profile files.

Listing 17 shows /etc/skel/.bashrc from the above system. This file might be different on a different release or different distribution, but it gives you an idea of how the default user setup can be done.

Listing 17. /etc/skel/.bashrc on Fedora 7

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
```

As you can see, the global /etc/bashrc is sourced, then any user specific instructions can be added. Listing 18 shows the part of /etc/bashrc in which the .sh scripts from /etc/profile.d are sourced.

Listing 18. Sourcing .sh scripts from /etc/profile.d

```
        for i in /etc/profile.d/*.sh; do
                if [ -r "$i" ]; then
                        . $i
        fi
        done
        unset i
```

Note that the variable, i, is unset after the loop.

Set command search paths

Your default profiles often include PATH variables for local functions or for products that you may have installed. You can set these in the skeleton files in /etc/skel, modify /etc/profile, /etc/bashrc, or create a file in /etc/profile.d if your system uses that. If you do modify the system files, be sure to check that your changes are intact after any system updates. Listing 19 shows how to add a new directory, /opt/productxyz/bin, to either the front or rear of your existing PATH.

Listing 19. Adding a path directory

```
PATH="$PATH${PATH:+:}/opt/productxyz/bin"
PATH="/opt/productxyz/bin${PATH:+:}$PATH"
```

Although not strictly required, the expression ${PATH:+:} inserts a path separator (colon) only if the PATH variable is unset or null.

Section 4. System log files

This section covers material for topic 1.111.3 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to configure and manage system logs, including these tasks:

- Manage the type and level of information logged
- Rotate and archive log files automatically
- Scan log files for notable activity
- Monitor log files
- Track down problems reported in log files

Manage the type and level of information logged

The system logging facility on a Linux system provides system logging and kernel message trapping. Logging can be done on a local system or sent to a remote system, and the level of logging can be finely controlled through the /etc/syslog.conf configuration file. Logging is performed by the `syslogd` daemon, which normally receives input through the /dev/log socket, as shown in Listing 20.

Listing 20. /dev/log is a socket

```
ian@pinguino:~$ ls -l /dev/log
srw-rw-rw- 1 root root 0 2007-07-05 15:42 /dev/log
```

For local logging, the main file is usually /var/log/messages, but many other files are used in most installations, and you can customize these extensively. For example, you may want a separate log for messages from the mail system.

The syslog.conf configuration file

The syslog.conf file is the main configuration file for the syslogd daemon. Logging is based on a combination of facility and priority. The defined facilities are auth (or security), authpriv, cron, daemon, ftp, kern, lpr, mail, mark, news, syslog, user, uucp, and local0 through local7. The keyword `auth` should be used instead of `security`, and the keyword `mark` is for internal use.

The priorities (in ascending order) are:

1. debug
2. info
3. notice
4. warning (or warn)
5. err (or error)
6. crit
7. alert
8. emerg (or panic)

The parenthesized keywords (warn, error, and panic) are now deprecated.

Entries in syslog.conf specify logging rules. Each rule has a selector field and an action field, which are separated by one or more spaces or tabs. The selector field identifies the facility and the priorities that the rule applies to, and the action field identifies the logging action for the facility and priorities. The default behavior is to take the action for the specified level and for all higher levels, although it is possible to limit logging to specific levels. Each selector consists of a facility and a priority separated by a period (dot). Multiple facilities for a given action can be specified by separating them with a comma. Multiple facility/priority pairs for a given action can be specified by separating them with a semi-colon. Listing 21 shows an example of a simple syslog.conf.

Listing 21. Example syslog.conf

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none
/var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*
-/var/log/maillog

# Log cron stuff
cron.*                                                  /var/log/cron
```

```
# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*
/var/log/boot.log
```

Notes:

- As with many configuration files, lines starting with # and blank lines are ignored.
- An * may be used to refer to all facilities or all priorities.
- The special priority keyword `none` indicates that no logging for this facility should be done with this action.
- The hyphen before a file name (such as -/var/log/maillog, in this example) indicates that the log file should not be synchronized after every write. You might lose information after a system crash, but you might gain performance by doing this.

The actions are generically referred to as "logfiles," although they do not have to be real files. Table 9 describe the possible logfiles.

Table 9. Actions in syslog.conf

Action	Purpose
Regular File	Specify the full pathname, beginning with a slash (/). Prefix it with a hyphen (-) to omit syncing the file after each log entry. This may cause information loss if a crash occurs, but may improve performance.
Named Pipes	A fifo or named pipe can be used as a destination for log messages by putting a pipe symbol (|) before the filename. You must create the fifo using the `mkfifo` command before starting (or restarting) `syslogd`. Fifos are sometimes used for debugging.
Terminal and Console	A terminal such as /dev/console.
Remote Machine	To forward messages to another host, put an at (@) sign before the hostname. Note that messages are not forwarded from the receiving host.
List of Users	A comma-separated list of users to receive a message (if the user is

	logged in). The root user is frequently included here.
Everyone logged on	Specify an asterisk (*) to have everyone logged on notified using the `wall` command.

You may prefix ! to a priority to indicate that the action should not apply to this level and higher. Similarly you may prefix it with = to indicate that the rule applies only to this level or with != to indicate that the rule applies to all except this level. Listing 22 shows some examples, and the man page for syslog.conf has many more examples.

Listing 22. More syslog.conf examples

```
# Store all kernel messages in /var/log/kernel.
# Send critical and higher ones to remote host pinguino and to the
console
# Finally, Send info, notice and warning messages to
/var/log/kernel-info
#
kern.*                          /var/log/kernel
kern.crit                       @pinguino
kern.crit                       /dev/console
kern.info;kern.!err             /var/log/kernel-info

# Store all mail messages except info priority in /var/log/mail.
mail.*;mail.!=info              /var/log/mail
```

Rotate and archive log files automatically

With the amount of logging that is possible, you need to be able to control the size of log files. This is done using the `logrotate` command, which is usually run as a cron job. Cron jobs are covered later in this tutorial in the section Scheduling jobs. The general idea behind the `logrotate` command is that log files are periodically backed up and a new log is started. Several generations of log are kept, and when a log ages to the last generation, it may be archived. For example, it might be mailed to an archival user.

You use the /etc/logrotate.conf configuration file to specify how your log rotating and archiving should happen. You can specify different frequencies, such as daily, weekly, or monthly, for different log files, and you can control the number of generations to keep and when or whether to mail copies to an archival user. Listing 23 shows a sample /etc/logrotate.conf file.

Listing 23. Sample /etc/logrotate.conf

```
# rotate log files weekly
weekly
```

```
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

# system-specific logs may be configured here
```

The logrotate.conf file has global options at the beginning. These are the defaults if nothing more specific is specified elsewhere. In this example, log files are rotated weekly, and four weeks worth of backups are kept. Once a log file is rotated, a new one is automatically created in place of the old one. Note that the logrotate.conf file may include specifications from other files. Here, all the files in /etc/logrotate.d are included.

This example also includes specific rules for /var/log/wtmp and /var/log/btmp, which are rotated monthly. No error message is issued if the files are missing. A new file is created, and only one backup is kept.

In this example, when a backup reaches the last generation, it is deleted because there is no specification of what else to do with it.

Note: The files /var/log/wtmp and /var/log/btmp record successful and unsuccessful login attempts, respectively. Unlike most log files, these are not clear text files. You may examine them using the `last` or `lastb` commands. See the man pages for these commands for details.

Log files may also be backed up when they reach a specific size, and commands may be scripted to run either prior to or after the backup operation. Listing 24 shows a more complex example.

Listing 24. Another logrotate configuration example

```
        /var/log/messages {
```

```
            rotate 5
            mail logsave@pinguino
            size 100k
             postrotate
                /usr/bin/killall -HUP syslogd
            endscript
        }
```

In this example, /var/log/messages is rotated after it reaches 100KB in size. Five backups are kept, and when the oldest backup ages out, it is mailed to logsave@pinguino. The `postrotate` introduces a script that restarts the syslogd daemon after the rotation is complete, by sending it the HUP signal. The `endscript` statement is required to terminate the script and is also required if a `prerotate` script is present. See the logrotate man page for more complete information.

Scan log files for notable activity

Log files entries are usually time stamped and contain the hostname of the reporting process, along with the process name. Listing 25 shows a few lines from /var/log/messages, containing entries from gconfd, ntpd, init, and yum.

Listing 25. Sample log file entries

```
Jul  5 15:28:24 lyrebird gconfd (root-2832): Exiting
Jul  5 15:31:06 lyrebird ntpd[2063]: synchronized to 87.98.219.90,
stratum 2
Jul  5 15:31:06 lyrebird ntpd[2063]: kernel time sync status change 0001
Jul  5 15:31:24 lyrebird init: Trying to re-exec init
Jul  5 15:31:24 lyrebird yum: Updated: libselinux.i386 2.0.14-2.fc7
Jul  5 15:31:24 lyrebird yum: Updated: libsemanage.i386 2.0.3-4.fc7
Jul  5 15:31:25 lyrebird yum: Updated: cups-libs.i386 1.2.11-2.fc7
Jul  5 15:31:25 lyrebird yum: Updated: libXfont.i386 1.2.9-2.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager.i386 0.6.5-7.fc7
Jul  5 15:31:27 lyrebird yum: Updated: NetworkManager-glib.i386
0.6.5-7.fc7
```

You can scan log files using a pager, such as `less`, or search for specific entries (such as kernel messages from host lyrebird) using `grep` as shown in Listing 26.

Listing 26. Scanning log files

```
[root@lyrebird ~]# less /var/log/messages
[root@lyrebird ~]# grep "lyrebird kernel" /var/log/messages | tail -n 9
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HCI socket layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP ver 2.8
Jul  5 15:26:46 lyrebird kernel: Bluetooth: L2CAP socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM socket layer
initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM TTY layer initialized
Jul  5 15:26:46 lyrebird kernel: Bluetooth: RFCOMM ver 1.8
```

```
Jul  5 15:26:46 lyrebird kernel: Bluetooth: HIDP (Human Interface
Emulation) ver 1.2
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized drm 1.1.0 20060810
Jul  5 15:26:59 lyrebird kernel: [drm] Initialized i915 1.6.0 20060119
on minor 0
```

Monitor log files

Occasionally you may need to monitor log files for events. For example, you might be trying to catch an infrequently occurring event at the time it happens. In such a case, you can use the `tail` command with the `-f` option to *follow* the log file. Listing 27 shows an example.

Listing 27. Following log file updates

```
[root@lyrebird ~]# tail -n 1 -f /var/log/messages
Jul  6 15:16:26 lyrebird syslogd 1.4.2: restart.
Jul  6 15:16:26 lyrebird kernel: klogd 1.4.2, log source = /proc/kmsg
started.
Jul  6 15:19:35 lyrebird yum: Updated: samba-common.i386 3.0.25b-2.fc7
Jul  6 15:19:35 lyrebird yum: Updated: procps.i386 3.2.7-14.fc7
Jul  6 15:19:36 lyrebird yum: Updated: samba-client.i386 3.0.25b-2.fc7
Jul  6 15:19:37 lyrebird yum: Updated: libsmbclient.i386 3.0.25b-2.fc7
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Received signal 15, shutting
down cleanly
Jul  6 15:19:46 lyrebird gconfd (ian-3267): Exiting
Jul  6 15:19:57 lyrebird yum: Updated: bluez-gnome.i386 0.8-1.fc7
```

Track down problems reported in log files

When you find problems in log files, you will want to note the time, the hostname, and the process that generated the problem. If the message identifies the problem specifically enough for you to resolve it, you are done. If not, you might need to update syslog.conf to specify that more messages be logged for the appropriate facility. For example, you might need to show informational messages instead of warning messages or even debug level messages. Your application may have additional facilities that you can use.

Finally, if you need to put marks in the log file to help you know what messages were logged at what stage of your debugging activity, you can use the `logger` command from a terminal window or shell script to send a message of your choice to the syslog daemon for logging according to the rules in syslog.conf.

Section 5. Scheduling jobs

This section covers material for topic 1.111.4 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

- Use the `cron` or `anacron` commands to run jobs at regular intervals
- Use the `at` command to run jobs at a specific time
- Manage cron and at jobs
- Configure user access to the cron and at services

In the previous section, you learned about the `logrotate` command and saw the need to run it periodically. You will see the same need to run commands regularly in the next two sections on backup and network time services. These are only some of the many administrative tasks that have to be done frequently and regularly. In this section, you learn about the tools that are used to automate periodic job scheduling and also the tools used to run a job at some specific time.

Run jobs at regular intervals

Running jobs at regular intervals is managed by the *cron* facility, which consists of the `crond` daemon and a set of tables describing what work is to be done and with what frequency. The daemon wakes up every minute and checks the crontabs to determine what needs to be done. Users manage crontabs using the `crontab` command. The `crond` daemon is usually started by the init process at system startup.

To keep things simple, let's suppose that you want to run the command shown in Listing 28 on a regular basis. This command doesn't actually do anything except report the day and the time, but it illustrates how to use `crontab` to set up cron jobs, and we'll know when it was run from the output. Setting up crontab entries requires a string with escaped shell metacharacters, so it is best done with simple commands and parameters, so in this example, the `echo` command will be run from within a script /home/ian/mycrontab.sh, which takes no parameters. This saves some careful work with escape characters.

Listing 28. A simple command example.

```
[ian@lyrebird ~]$ cat mycrontest.sh
#!/bin/bash
 echo "It is now $(date +%T) on $(date +%A)"
[ian@lyrebird ~]$ ./mycrontest.sh
It is now 18:37:42 on Friday
```

Creating a crontab

To create a crontab, you use the `crontab` command with the `-e` (for "edit") option. This will open the `vi` editor unless you have specified another editor in the EDITOR or VISUAL environment variable.

Each crontab entry contains six fields:

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week
6. String to be executed by `sh`

Minutes and hours range from 0-59 and 0-12, respectively, while day or month and month of year range from 1-31 and 1-12, respectively. Day of week ranges from 0-6 with 0 being Sunday. Day of week may also be specified as sun, mon, tue, and so on. The sixth field is everything after the fifth field, is interpreted as a string to pass to `sh`. A percent sign (%) will be translated to a newline, so if you want a % or any other special character, precede it with a backslash (\). The line up to the first % is passed to the shell, while any line(s) after the % are passed as standard input.

The various time-related fields can specify an individual value, a range of values, such as 0-10 or sun-wed, or a comma-separated list of individual values and ranges. So a somewhat artificial crontab entry for our example command might be as shown in Listing 29.

Listing 29. A simple crontab example.

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
```

In this example, our command is executed at the 0th, 20th, and 40th minutes (every 20 minutes), for the hours between 10 P.M. and midnight on Fridays and Saturdays during July. See the man page for crontab(5) for details on additional ways to specify times.

What about the output?

You may be wondering what happens to any output from the command. Most

commands designed for use with the cron facility will log output using the syslog facility that you learned about in the previous section. However any output that is directed to stdout will be mailed to the user. Listing 30 shows the output you might receive from our example command.

Listing 30. Mailed cron output

```
From ian@lyrebird.raleigh.ibm.com  Fri Jul  6 23:00:02 2007
Date: Fri, 6 Jul 2007 23:00:01 -0400
From: root@lyrebird.raleigh.ibm.com (Cron Daemon)
To: ian@lyrebird.raleigh.ibm.com
Subject: Cron <ian@lyrebird> /home/ian/mycrontest.sh
Content-Type: text/plain; charset=UTF-8
Auto-Submitted: auto-generated
X-Cron-Env: <SHELL=/bin/sh>
X-Cron-Env: <HOME=/home/ian>
X-Cron-Env: <PATH=/usr/bin:/bin>
X-Cron-Env: <LOGNAME=ian>
X-Cron-Env: <USER=ian>

It is now 23:00:01 on Friday
```

Where is my crontab?

The crontab that you created with the `crontab` command is stored in /etc/spool/cron under the name of the user who created it. So the above crontab is stored in /etc/spool/cron/ian. Given this, you will not be surprised to learn that the `crontab` command, like the `passwd` command you learned about earlier, is an suid program that runs with root authority.

/etc/crontab

In addition to the user crontab files in /var/spool/cron, `cron` also checks /etc/crontab and files in the /etc/cron.d directory. These system crontabs have one additional field between the fifth time entry (day) and the command. This additional field specifies the user for whom the command should be run, normally root. A /etc/crontab might look like the example in Listing 31.

Listing 31. /etc/crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

In this example, the real work is done by the `run-parts` command, which runs

scripts from /etc/cron.hourly, /etc/cron.daily, and so on; /etc/crontab simply controls the timing of the recurring jobs. Note that the commands here all run as root. Note also that the crontab can include shell variables assignments that will be set before the commands are run.

Anacron

The cron facility works well for systems that run continuously. For systems that may be turned off much of the time, such as laptops, another facility, the *anacron* (for "anachronistic cron") can handle scheduling of the jobs usually done daily, weekly, or monthly by the cron facility. Anacron does not handle hourly jobs.

Anacron keeps timestamp files in /var/spool/anacron to record when jobs are run. When anacron runs, it checks to see if the required number of days has passed since the job was last run and runs it if necessary. The table of jobs for anacron is stored in /etc/anacrontab, which has a slightly different format than /etc/crontab. As with /etc/crontab, /etc/anacrontab may contain environment settings. Each job has four fields.

1. period
2. delay
3. job-identifier
4. command

The period is a number of days, but may be specified as @monthly to ensure that a job runs only once per month, regardless of the number of days in the month. The delay is the number of minutes to wait after the job is due to run before actually starting it. You can use this to prevent a flood of jobs when a system first starts. The job identifier can contain any non-blank character except slashes (/).

Both /etc/crontab and /etc/anacrontab are updated by direct editing. You do not use the `crontab` command to update these files or files in the /etc/cron.d directory.

Run jobs at specific times

Sometimes you may need to run a job just once, rather than regularly. For this you use the `at` command. The commands to be run are read from a file specified with the `-f` option, or from stdin if `-f` is not used. The `-m` option sends mail to the user even if there is no stdout from the command. The `-v` option will display the time at which the job will run before reading the job. The time is also displayed in the output. Listing 32 shows an example of running the `mycrontest.sh` script that you used earlier. Listing 33 shows the output that is mailed back to the user after the job runs.

Notice that it is somewhat more compact than the corresponding output from the cron job.

Listing 32. Using the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh -v 10:25
Sat Jul  7 10:25:00 2007

job 5 at Sat Jul  7 10:25:00 2007
```

Listing 33. Job output from at

```
From ian@lyrebird.raleigh.ibm.com  Sat Jul  7 10:25:00 2007
Date: Sat, 7 Jul 2007 10:25:00 -0400
From: Ian Shields <ian@lyrebird.raleigh.ibm.com>
Subject: Output from your job        5
To: ian@lyrebird.raleigh.ibm.com

It is now 10:25:00 on Saturday
```

Time specifications can be quite complex. Listing 34 shows a few examples. See the man page for `at` or the file /usr/share/doc/at/timespec or a file such as /usr/share/doc/at-3.1.10/timespec, where 3.1.10 in this example is the version of the at package.

Listing 34. Time values with the at command

```
[ian@lyrebird ~]$ at -f mycrontest.sh  10pm tomorrow
job 14 at Sun Jul  8 22:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 tuesday
job 15 at Tue Jul 10 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 july 11
job 16 at Wed Jul 11 02:00:00 2007
[ian@lyrebird ~]$ at -f mycrontest.sh 2:00 next week
job 17 at Sat Jul 14 02:00:00 2007
```

The `at` command also has a `-q` option. Increasing the queue increases the nice value for the job. There is also a `batch` command, which is similar to the `at` command except that jobs are run only when the system load is low enough. See the man pages for more details on these features.

Manage scheduled jobs

Listing scheduled jobs

You can manage your cron and at jobs. Use the `crontab` command with the `-l` option to list your crontab, and use the `atq` command to display the jobs you have queued using the `at` command, as shown in Listing 35.

Listing 35. Displaying scheduled jobs

```
[ian@lyrebird ~]$ crontab -l
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
```

If you want to review the actual command scheduled for execution by `at`, you can use the `at` command with the `-c` option and the job number. You will notice that most of the environment that was active at the time the `at` command was issued is saved with the scheduled job. Listing 36 shows part of the output for job 15.

Listing 36. Using at -c with a job number

```
#!/bin/sh
# atrun uid=500 gid=500
# mail ian 0
umask 2
HOSTNAME=lyrebird.raleigh.ibm.com; export HOSTNAME
SHELL=/bin/bash; export SHELL
HISTSIZE=1000; export HISTSIZE
SSH_CLIENT=9.67.219.151\ 3210\ 22; export SSH_CLIENT
SSH_TTY=/dev/pts/5; export SSH_TTY
USER=ian; export USER
 ...
HOME=/home/ian; export HOME
LOGNAME=ian; export LOGNAME
 ...
cd /home/ian || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
${SHELL:-/bin/sh} << `(dd if=/dev/urandom count=200 bs=1 \
   2>/dev/null|LC_ALL=C tr -d -c '[:alnum:]')`

#!/bin/bash
 echo "It is now $(date +%T) on $(date +%A)"
```

Note that the contents of our script file have been copied in as a here-document that will be executed by the shell specified by the SHELL variable or /bin/sh if the SHELL variable is not set. See the tutorial LPI exam 101 prep, Topic 103: GNU and UNIX commands if you need to review here-documents.

Deleting scheduled jobs

You can delete all scheduled cron jobs using the `cron` command with the `-r` option as illustrated in Listing 37.

Listing 37. Displaying and deleting cron jobs

```
[ian@lyrebird ~]$ crontab -l
```

```
0,20,40 22-23 * 7 fri-sat /home/ian/mycrontest.sh
[ian@lyrebird ~]$ crontab -r
[ian@lyrebird ~]$ crontab -l
no crontab for ian
```

To delete system cron or anacron jobs, edit /etc/crontab, /etc/anacrontab, or edit or delete files in the /etc/cron.d directory.

You can delete one or more jobs that were scheduled with the `at` command by using the `atrm` command with the job number. Multiple jobs should be separated by spaces. Listing 38 shows an example.

Listing 38. Displaying and removing jobs with atq and atrm

```
[ian@lyrebird ~]$ atq
16      Wed Jul 11 02:00:00 2007 a ian
17      Sat Jul 14 02:00:00 2007 a ian
14      Sun Jul  8 22:00:00 2007 a ian
15      Tue Jul 10 02:00:00 2007 a ian
[ian@lyrebird ~]$ atrm 16 14 15
[ian@lyrebird ~]$ atq
17      Sat Jul 14 02:00:00 2007 a ian
```

Configure user access to job scheduling

If the file /etc/cron.allow exists, any non-root user must be listed in it in order to use `crontab` and the cron facility. If /etc/cron.allow does not exist, but /etc/cron.deny does exist, a non-root user who is listed in it cannot use `crontab` or the cron facility. If neither of these files exists, only the super user will be allowed to use this command. An empty /etc/cron.deny file allows all users to use the cron facility and is the default.

The corresponding /etc/at.allow and /etc/at.deny files have similar effects for the at facility.

Section 6. Data backup

This section covers material for topic 1.111.5 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 3.

In this section, learn how to:

- Plan a backup strategy

- Dump a raw device to a file or restore a raw device from a file
- Perform partial and manual backups
- Verify the integrity of backup files
- Restore filesystems partially or fully from backups

Plan a backup strategy

Having a good backup is a necessary part of system administration, but deciding what to back up and when and how can be complex. Databases, such as customer orders or inventory, are usually critical to a business and many include specialized backup and recovery tools that are beyond the scope of this tutorial. At the other extreme, some files are temporary in nature and no backup is needed at all. In this section, we focus on system files and user data and discuss some of the considerations, methods, and tools for backup of such data.

There are three general approaches to backup:

1. A *full* backup is a complete backup, usually of a whole filesystem, directory, or group of related files. This takes the longest time to create, so it is usually used with one of the other two approaches.

2. A *differential* or *cumulative* backup is a backup of all things that have changed since the last full backup. Recovery requires the last full backup plus the latest differential backup.

3. An *incremental* backup is a backup of only those changes since the last incremental backup. Recovery requires the last full backup plus all of the incremental backups (in order) since the last full backup.

What to back up

When deciding what to back up, you should consider how volatile the data is. This will help you determine how often it should be backed up. Similarly, critical data should be backed up more often than non-critical data. Your operating system will probably be relatively easy to rebuild, particularly if you use a common image for several systems, although the files that customize each system would be more important to back up.

For programming staff, it may be sufficient to keep backups of repositories such as CVS repositories, while individual programmers' sandboxes may be less important. Depending on how important mail is to your operation, it may suffice to have infrequent mail backups, or it may be necessary to be able to recover mail to the

most recent date possible. You may want to keep backups of system cron files, but may not be so concerned about scheduled jobs for individual users.

The Filesystem Hierarchy Standard provides a classification of data that may help you with your backup choices. For details, see the tutorial LPI exam 101 prep: Devices, Linux filesystems, and the Filesystem Hierarchy Standard.

Once you have decided what to back up, you need to decide how often to do a full backup and whether to do differential or incremental backups in between those full backups. Having made those decisions, the following suggestions will help you choose appropriate tools.

Automating backups

In the previous section, you learned how to schedule jobs, and the cron facility is ideal for helping to automate the scheduling of your backups. However, backups are frequently made to removable media, particularly tape, so operator intervention is probably going to be needed. You should create and use backup scripts to ensure that the backup process is as automatic and repeatable as possible.

Dump and restore raw devices

One way to make a full backup of a filesystem is to make an image of the partition on which it resides. A *raw device*, such as /dev/hda1 or /dev/sda2, can be opened and read as a sequential file. Similarly, it can be written from a backup as a sequential file. This requires no knowledge on the part of the backup tool as to the filesystem layout, but does require that the restore be done to space that is at least as large as the original. Some tools that handle raw devices are *filesystem aware*, meaning that they understand one or more of the Linux filesystems. These utilities can dump from a raw device but do not dump unused parts of the partition. They may or may not require restoration to the same or larger sized partition. The `dd` command is an example of the first type, while the `dump` command is an example of the second type that is specific to the ext2 and ext3 filesystems.

The dd command

In its simplest form, the `dd` command copies an input file to an output file, where either file may be a raw device. For backing up a raw device, such as /dev/hda1 or /dev/sda2, the input file would be a raw device. Ideally, the filesystem on the device should be unmounted, or at least mounted read only, to ensure that data does not change during the backup. Listing 39 shows an example.

Listing 39. Backup a partition using dd

```
[root@lyrebird ~]# dd if=/dev/sda3 of=backup-1
```

```
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 49.3103 s, 21.2 MB/s
```

The `if` and `of` parameters specify the input and output files respectively. In this example, the input file is a raw device, dev/sda3, and the output file is a file, backup-1, in the root user's home directory. To dump the file to tape or floppy disk, you would specify something like `of=/dev/fd0` or `of=/dev/st0`.

Note that 1,044,610,560 bytes of data was copied and the output file is indeed that large, even though only about 3% of this particular partition is actually used. Unless you are copying to a tape with hardware compression, you will probably want to compress the data. Listing 40 shows one way to accomplish this, along with the output of `ls` and `df` commands, which show you the file sizes and the usage percentage of the filesystem on /dev/sda3.

Listing 40. Backup with compression using dd

```
[root@lyrebird ~]# dd if=/dev/sda3 | gzip > backup-2
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 117.723 s, 8.9 MB/s
[root@lyrebird ~]# ls -l backup-[12]
-rw-r--r-- 1 root root 1044610560 2007-07-08 15:17 backup-1
-rw-r--r-- 1 root root  266932272 2007-07-08 15:56 backup-2
[root@lyrebird ~]# df -h /dev/sda3
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda3             972M   28M  944M   3% /grubfile
```

The gzip compression reduced the file size to about 20% of the uncompressed size. However, unused blocks may contain arbitrary data, so even the compressed backup may be much larger than the total data on the partition.

If you divide the size by the number of records processed by `dd`, you will see that `dd` is writing 512-byte blocks of data. When copying to a raw output device such as tape, this can result in a very inefficient operation, so `dd` can read or write data in much larger blocks. Specify the `obs` option to change the output size or the `ibs` option to specify the input block size. You can also specify just `bs` to set both input and output block sizes to a common value.

If you need multiple tapes or other removable storage to store your backup, you will need to break it into smaller pieces using a utility such as `split`.

If you need to skip blocks such as disk or tape labels, you can do so with `dd`. See the man page for examples.

Besides just copying data, the `dd` command can do several conversions, such as between ASCII and EBCDIC, between big-endian and little-endian, or between variable-length data records and fixed-length data records. Obviously these

conversions are likely to be useful when copying real files rather than raw devices. Again, see the man page for details.

The dump command

The dump command can be used for full, differential, or incremental backups on ext2 or ext3 filesystems. Listing 41 shows an example.

Listing 41. Backup with compression using dump

```
[root@lyrebird ~]# dump -0 -f backup-4 -j -u /dev/sda3
  DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
  DUMP: Dumping /dev/sda3 (/grubfile) to backup-4
  DUMP: Label: GRUB
  DUMP: Writing 10 Kilobyte records
  DUMP: Compressing output at compression level 2 (bzlib)
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 12285 blocks.
  DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:47:48 2007
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: Closing backup-4
  DUMP: Volume 1 completed at: Sun Jul  8 16:47:57 2007
  DUMP: Volume 1 took 0:00:09
  DUMP: Volume 1 transfer rate: 819 kB/s
  DUMP: Volume 1 12260kB uncompressed, 7377kB compressed, 1.662:1
  DUMP: 12260 blocks (11.97MB) on 1 volume(s)
  DUMP: finished in 9 seconds, throughput 1362 kBytes/sec
  DUMP: Date of this level 0 dump: Sun Jul  8 16:47:47 2007
  DUMP: Date this dump completed:  Sun Jul  8 16:47:57 2007
  DUMP: Average transfer rate: 819 kB/s
  DUMP: Wrote 12260kB uncompressed, 7377kB compressed, 1.662:1
  DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[2-4]
-rw-r--r-- 1 root root 266932272 2007-07-08 15:56 backup-2
-rw-r--r-- 1 root root 266932272 2007-07-08 15:44 backup-3
-rw-r--r-- 1 root root   7554939 2007-07-08 16:47 backup-4
```

In this example, `-0` specifies the *dump level* which is an integer, historically from 0 to 9, where 0 specifies a full dump. The `-f` option specifies the output file, which may be a raw device. Specify `-` to direct the output to stdout. The `-j` option specifies compression, with a default level of 2, using bzlib compression. You can use the `-z` option to specify zlib compression if you prefer. The `-u` option causes the record of dump information, normally /etc/dumpdates, to be updated. Any parameters after the options represent a file or list of files, where the file may also be a raw device, as in this example. Notice how much smaller the backup is when the backup program is aware of the filesystem structure and can avoid the saving of unused blocks on the device.

If output is to a device such as tape, the dump command will prompt for another volume as each volume is filled. You can also provide multiple file names separated by commas. For example, if you wanted an unattended dump that required two tapes, you could load the tapes on /dev/st0 and /dev/st1, schedule the dump command specifying both tapes as output, and go home to sleep.

When you specify a dump level greater than 0, an incremental dump is performed of all files that are new or have changed since the last dump at a lower level was taken. So a dump at level 1 will be a differential dump, even if a dump at level 2 or higher has been taken in the meantime. Listing 42 shows the result of updating the time stamp of an existing file on /dev/sda3 and creating a new file, then taking a dump at level 2. After that, another new file is created and a dump at level 1 is taken. The information from /etc/dumpdates is also shown. For brevity, part of the second dump output has been omitted.

Listing 42. Backup with compression using dump

```
[root@lyrebird ~]# dump -2 -f backup-5 -j -u /dev/sda3
  DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
  DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
  DUMP: Dumping /dev/sda3 (/grubfile) to backup-5
  DUMP: Label: GRUB
  DUMP: Writing 10 Kilobyte records
  DUMP: Compressing output at compression level 2 (bzlib)
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 91 blocks.
  DUMP: Volume 1 started with block 1 at: Sun Jul  8 16:55:47 2007
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: Closing backup-5
  DUMP: Volume 1 completed at: Sun Jul  8 16:55:47 2007
  DUMP: 90 blocks (0.09MB) on 1 volume(s)
  DUMP: finished in less than a second
  DUMP: Date of this level 2 dump: Sun Jul  8 16:55:46 2007
  DUMP: Date this dump completed:  Sun Jul  8 16:55:47 2007
  DUMP: Average transfer rate: 0 kB/s
  DUMP: Wrote 90kB uncompressed, 15kB compressed, 6.000:1
  DUMP: DUMP IS DONE
[root@lyrebird ~]# echo "This data is even newer" >/grubfile/newerfile
[root@lyrebird ~]# dump -1 -f backup-6 -j -u -A backup-6-toc /dev/sda3
  DUMP: Date of this level 1 dump: Sun Jul  8 17:08:18 2007
  DUMP: Date of last level 0 dump: Sun Jul  8 16:47:47 2007
  DUMP: Dumping /dev/sda3 (/grubfile) to backup-6
    ...
  DUMP: Wrote 100kB uncompressed, 16kB compressed, 6.250:1
  DUMP: Archiving dump to backup-6-toc
  DUMP: DUMP IS DONE
[root@lyrebird ~]# ls -l backup-[4-6]
-rw-r--r-- 1 root root 7554939 2007-07-08 16:47 backup-4
-rw-r--r-- 1 root root   16198 2007-07-08 16:55 backup-5
-rw-r--r-- 1 root root   16560 2007-07-08 17:08 backup-6
[root@lyrebird ~]# cat /etc/dumpdates
/dev/sda3 0 Sun Jul  8 16:47:47 2007 -0400
/dev/sda3 2 Sun Jul  8 16:55:46 2007 -0400
/dev/sda3 1 Sun Jul  8 17:08:18 2007 -0400
```

Notice that backup-6 is, indeed, larger than backup 5. The level 1 dump illustrates the use of the `-A` option to create a table of contents that can be used to determine if a file is on an archive without actually mounting the archive. This is particularly useful with tape or other removable archive volumes. You will see these examples again when we discuss restoring data later in this section.

The `dump` command can dump files or subdirectories, but you cannot update

/etc/dumpdates and only level 0, of full dump, is supported.

Listing 43 illustrates the dump command dumping a directory, /usr/include/bits, and its contents to floppy disk. In this case, the dump will not fit on a single floppy, so a new volume is required. The prompt and response are shown in bold.

Listing 43. Backup a directory to multiple volumes using dump

```
[root@lyrebird ~]# dump -0 -f /dev/fd0 /usr/include/bits
  DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
  DUMP: Dumping /dev/sdb9 (/ (dir usr/include/bits)) to /dev/fd0
  DUMP: Label: /
  DUMP: Writing 10 Kilobyte records
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 2790 blocks.
  DUMP: Volume 1 started with block 1 at: Mon Jul  9 16:03:30 2007
  DUMP: dumping (Pass III) [directories]
  DUMP: End of tape detected
  DUMP: Closing /dev/fd0
  DUMP: Volume 1 completed at: Mon Jul  9 16:04:49 2007
  DUMP: Volume 1 1470 blocks (1.44MB)
  DUMP: Volume 1 took 0:01:19
  DUMP: Volume 1 transfer rate: 18 kB/s
  DUMP: Change Volumes: Mount volume #2
  DUMP: Is the new volume mounted and ready to go?: ("yes" or "no") y
  DUMP: Volume 2 started with block 1441 at: Mon Jul  9 16:05:10 2007
  DUMP: Volume 2 begins with blocks from inode 2
  DUMP: dumping (Pass IV) [regular files]
  DUMP: Closing /dev/fd0
  DUMP: Volume 2 completed at: Mon Jul  9 16:06:28 2007
  DUMP: Volume 2 1410 blocks (1.38MB)
  DUMP: Volume 2 took 0:01:18
  DUMP: Volume 2 transfer rate: 18 kB/s
  DUMP: 2850 blocks (2.78MB) on 2 volume(s)
  DUMP: finished in 109 seconds, throughput 26 kBytes/sec
  DUMP: Date of this level 0 dump: Mon Jul  9 16:03:23 2007
  DUMP: Date this dump completed:  Mon Jul  9 16:06:28 2007
  DUMP: Average transfer rate: 18 kB/s
  DUMP: DUMP IS DONE
```

If you back up to tape, remember that the tape will usually be rewound after each job. Devices with a name like /dev/st0 or /dev/st1 automatically rewind. The corresponding non-rewind equivalent devices are /dev/nst0 and /dev/nst1. In any event, you can always use the `mt` command to perform magnetic tape operations such as forward spacing over files and records, back spacing, rewinding, and writing EOF marks. See the man pages for `mt` and `st` for additional information.

If you select the dump levels judiciously, you can minimize the number of archives you need to restore to any particular level. See the man pages for `dump` for a suggestion based on the Towers of Hanoi puzzle.

As with the `dd` command, there are many options that are not covered in this brief introduction. See the man pages for more details.

Partial and manual backups

So far, you have learned about tools that work well for backing up whole filesystems. Sometimes your backup needs to target selected files or subdirectories without backing up the whole filesystem. For example, you might need a weekly backup of most of your system, but daily backups of your mail files. Two other programs, `cpio` and `tar`, are more commonly used for this purpose. Both can write archives to files or to devices such as tape or floppy disk, and both can restore from such archives. Of the two, `tar` is more commonly used today, possibly because it handles complete directories better, and GNU tar supports both gzip and bzip compression.

Using cpio

The `cpio` command operates in *copy-out* mode to create an archive, *copy-in* mode to restore an archive, or *copy-pass* mode to copy a set of files from one location to another. You use the `-o` or `--create` option for copy-out mode, the `-i` or `--extract` option for copy-in mode, and the `-p` or `--pass-through` option for copy-pass mode. Input is a list of files provided on stdin. Output is either to stdout or to a device or file specified with the `-f` or `--file` option.

Listing 44 shows how to generate a list of files using the `find` command. Note the use of the `-print0` option on `find` to generate null-terminate strings for file names, and the corresponding `--null` option on `cpio` to read this format. This will correctly handle file names that have embedded blank or newline characters.

Listing 44. Back up a home directory using cpio

```
[root@lyrebird ~]#  find ~ian -depth -print0 | cpio --null -o
>backup-cpio-1
18855 blocks
```

If you'd like to see the files listed as they are archived, add the `-v` option to `cpio`.

As with other commands that can archive to tape, the block size may be specified. For details on this and other options, see the man page.

Using tar

The `tar` (originally from *Tape ARchive*) creates an archive file, or *tarfile* or *tarball*, from a set of input files or directories; it also restores files from such an archive. If a directory is given as input to `tar`, all files and subdirectories are automatically included, which makes `tar` very convenient for archiving subtrees of your directory structure.

As with the other archiving commands we have discussed, output can be to a file, a

device such as tape or diskette, or stdout. The output location is specified with the `-f` option. Other common options are `-c` to create an archive, `-x` to extract an archive, `-v` for verbose output, which lists the files being processed, `-z` to use gzip compression, and `-j` to use bzip2 compression. Most `tar` options have a short form using a single hyphen and a long form using a pair of hyphens. The short forms are illustrated here. See the man pages for the long form and for additional options.

Listing 45 shows how to create a backup of the system cron jobs using `tar`.

Listing 45. Backup of system cron jobs using tar

```
[root@lyrebird ~]# tar -czvf backup-tar-1 /etc/*crontab /etc/cron.d
tar: Removing leading `/' from member names
/etc/anacrontab
/etc/crontab
/etc/cron.d/
/etc/cron.d/sa-update
/etc/cron.d/smolt
```

In the first line of output, you are told that `tar` will remove the leading slash (/) from member names. This allows files to be restored to some other location for verification before replacing system files. It is a good idea to avoid mixing absolute path names with relative path names when creating an archive, since all will be relative when restoring from the archive.

The `tar` command can append additional files to an archive using the `-r` or `--append` option. This may cause multiple copies of a file in the archive. In such a case, the *last* one will be restored during a restore operation. You can use the `--occurrence` option to select a specific file among multiples. If the archive is on a regular filesystem instead of tape, you may use the `-u` or `--update` option to update an archive. This works like appending to an archive, except that the time stamps of the files in the archive are compared with those on the filesystem, and only files that have been modified since the archived version are appended. As mentioned, this does not work for tape archives.

As with the other commands you have studied here, there are many options that are not covered in this brief introduction. See the man or info pages for more details.

Backup file integrity

Backup file integrity is extremely important. There is no point in having a backup if it is bad. A good backup strategy also involves checking your backups.

The first step to ensuring backup integrity is to ensure that you have properly captured the data you are backing up. If the filesystem is unmounted or mounted read only, this is usually straightforward as the data you are backing up cannot

change during your backup. If you must back up filesystems, directories, or files that are subject to modification while you are taking the backup, you should verify that no changes have been made during your backup. If changes were made, you will need to have a strategy for capturing them, either by repeating the backup, or perhaps by replacing or superseding the affected files in your backup. Needless to say, this will also affect your restore procedures.

Assuming you took good backups, you will periodically need to verify your backups. One way is to restore the backup to a spare volume and verify that it matches what you backed up. This is easiest to do right before you allow updates on the filesystem you are backing up. If you back up to media such as CD or DVD, you may be able to use the `diff` command as part of your backup procedure to ensure that your backup is good. Remember that even good backups can deteriorate in storage, so you should check periodically, even if you do verify at the time of backup. Keeping digests using programs such as `md5sum` or `sha1sum` is also a good check on the integrity of a backup file.

Restore filesystems from backups

A counterpart to backing up files is the ability to restore them when needed. Occasionally you will want to restore an entire filesystem, but it is far more common to need to restore only specific files or perhaps a set of directories. Almost always you will restore to some temporary space and verify that what you have restored is indeed what you want and is consistent with the current state of your system before actually making the restored files live.

A related issue is the need to verify that the items you want happen to be on a particular backup, as often happens when a user needs access to a version of a file that was modified or perhaps deleted "sometime in the last week or two." With these thoughts in mind, let's look at some of the restoration options.

Restoring a dd archive

Recall that the `dd` command was not filesystem aware, so you will need to restore a dump of a partition to find out what is on it. Listing 46 shows how to restore the partition that was dumped back in Listing 39 to a partition, /dev/sdc7, that was specially created on a removable USB drive just for this purpose.

Listing 46. Restoring a partition using dd

```
[root@lyrebird ~]# dd if=backup-1 of=/dev/sdc7
2040255+0 records in
2040255+0 records out
1044610560 bytes (1.0 GB) copied, 44.0084 s, 23.7 MB/s
```

Recall that we added some files to the filesystem on /dev/sda3 after this backup was taken. If you mount the newly restore partition and compare it with the original, you will see that this is indeed the case, as shown in Listing 47. Note that the file whose timestamp was updated using `touch` is not shown here, as you would expect.

Listing 47. Comparing the restored partition with current state

```
[root@lyrebird ~]# mount /dev/sdc7 /mnt/temp-dd/
[root@lyrebird ~]# diff -rq /grubfile/ /mnt/temp-dd/
Only in /grubfile/: newerfile
Only in /grubfile/: newfile
```

Restoring a dump archive using restore

Recall that our final use of `dump` was a differential backup and that we created a table of contents. Listing 48 shows how to use `restore` to check the files in the archive created by `dump`, using the archive itself (backup-5) or the table of contents (backup-6-toc).

Listing 48. Checking the contents of archives

```
[root@lyrebird ~]# restore -t -f backup-5
Dump tape is compressed.
Dump   date: Sun Jul  8 16:55:46 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 2 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
         2      .
    100481      ./ibshome
    100482      ./ibshome/index.html
        16      ./newfile
[root@lyrebird ~]# restore -t -A backup-6-toc
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
Starting inode numbers by volume:
        Volume 1: 2
         2      .
    100481      ./ibshome
    100482      ./ibshome/index.html
        16      ./newfile
        17      ./newerfile
```

The `restore` command can also compare the contents of an archive with the contents of the filesystem using the `-C` option. In Listing 49 we updated newerfile and then compared the backup with the filesystem.

Listing 49. Comparing an archive with a filesystem using restore

```
[root@lyrebird ~]# echo "something different" >/grubfile/newerfile
[root@lyrebird ~]# restore -C -f backup-6
Dump tape is compressed.
```

```
Dump   date: Sun Jul  8 17:08:18 2007
Dumped from: Sun Jul  8 16:47:47 2007
Level 1 dump of /grubfile on lyrebird.raleigh.ibm.com:/dev/sda3
Label: GRUB
filesys = /grubfile
./newerfile: size has changed.
Some files were modified!  1 compare errors
```

The `restore` command can restore interactively or automatically. Listing 50 shows how to restore newerfile to root's home directory (so you could examine it before replacing the updated file if needed), then replace the updated file with the backup copy. This example illustrates interactive restoration.

Listing 50. Restoring a file using restore

```
[root@lyrebird ~]# restore -i -f backup-6
Dump tape is compressed.
restore > ?
Available commands are:
        ls [arg] - list directory
        cd arg - change directory
        pwd - print current directory
        add [arg] - add `arg' to list of files to be extracted
        delete [arg] - delete `arg' from list of files to be extracted
        extract - extract requested files
        setmodes - set modes of requested directories
        quit - immediately exit program
        what - list dump header information
        verbose - toggle verbose flag (useful with ``ls'')
        prompt - toggle the prompt display
        help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls new*
newerfile
newfile
restore > add newerfile
restore > extract
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 # (none if no more volumes): 1
set owner/mode for '.'? [yn] y
restore > q
[root@lyrebird ~]# mv -f newerfile /grubfile
```

Restoring a cpio archive

The `cpio` command in copy-in mode (option `-i` or `--extract`) can list the contents of an archive or restore selected files. When you list the files, specifying the `--absolute-filenames` option reduces the number of extraneous messages that `cpio` will otherwise issue as it strips any leading / characters from each path that has one. Partial output from listing our previous archive is shown in Listing 51.

Listing 51. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio  -id --list --absolute-filenames <backup-cpio-1
```

```
/home/ian/.gstreamer-0.10/registry.i686.xml
/home/ian/.gstreamer-0.10
/home/ian/.Trash/gnome-terminal.desktop
/home/ian/.Trash
/home/ian/.bash_profile
```

Listing 52 shows how to restore all the files with "samp" in their path name or file name. The output has been piped through `uniq` to reduce the number of "Removing leading '/' ..." messages. You must specify the `-d` option to create directories; otherwise, all files are created in the current directory. Furthermore, `cpio` will not replace any newer files on the filesystem with archive copies unless you specify the `-u` or `--unconditional` option.

Listing 52. Restoring selected files using cpio

```
[root@lyrebird ~]# cpio  -ivd "*samp*" < backup-cpio-1 2>&1 |uniq
cpio: Removing leading `/' from member names
home/ian/crontab.samp
cpio: Removing leading `/' from member names
home/ian/sample.file
cpio: Removing leading `/' from member names
18855 blocks
```

Restoring a tar archive

The `tar` command can also compare archives with the current filesystem as well as restore files from archives. Use the `-d`, `--compare`, or `--diff` option to perform comparisons. The output will show files whose contents differ as well as files whose time stamps differ. Listing 53 shows verbose output (using option `-v`), from a comparison of the file created earlier and the files in /etc after /etc/crontab has been touched to alter its time stamp. The option `directory /` instructs `tar` to perform the comparison starting from the root directory rather than the current directory.

Listing 53. Comparing archives and files using tar

```
[root@lyrebird ~]# touch /etc/crontab
[root@lyrebird ~]# tar --diff -vf backup-tar-1 --directory /
etc/anacrontab
etc/crontab
etc/crontab: Mod time differs
etc/cron.d/
etc/cron.d/sa-update
etc/cron.d/smolt
```

Listing 54 shows how to extract just /etc/crontab and /etc/anacrontab into the current directory.

Listing 54. Extracting archive files using tar

```
[root@lyrebird ~]# tar -xzvf backup-tar-1 "*tab"
```

```
etc/anacrontab
etc/crontab
```

Note that `tar`, in contrast to `cpio` creates the directory hierarchy for you automatically.

The next section of this tutorial shows you how to maintain system time.

Section 7. System time

This section covers material for topic 1.111.6 for the Junior Level Administration (LPIC-1) exam 102. The topic has a weight of 4.

In this section, learn how to:

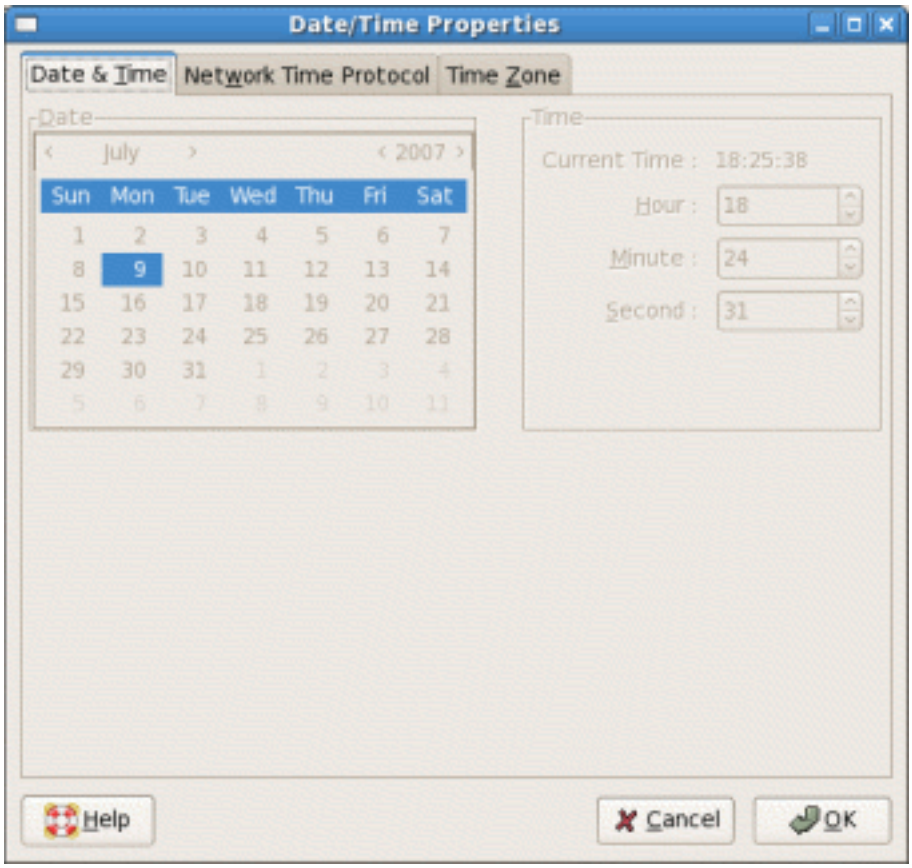
- Set the system date and time
- Set the BIOS clock to the correct UTC time
- Configure your time zone
- Configure the Network Time Protocol (NTP) service, including correcting for clock drift

Set the system date and time

System time on a Linux system is very important. You saw earlier how the cron and anacron facilities do things based on time, so they need an accurate time to base decisions on. Most of the backup and restore tools discussed in the previous section, along with development tools such as `make`, also depend on reliable time measurements. Most computers built since around 1980 include some kind of clock mechanism, and most built since 1984 or so have a persistent clock mechanism that keeps time even if the computer is turned off.

If you installed a Linux system graphically, you probably set the clock and chose a time zone suitable for your needs. You may have elected to use the Network Time Protocol (NTP) to set your clock, and you may or may not have elected to keep the system clock using Coordinated Universal Time or UTC. If you subsequently went to set the clock using graphical tools on a Fedora or Red Hat or similar system, you may have seen a dialog box like that in Figure 3.

Figure 3. Updating the date and time



Surprise! You can't actually set the clock yourself using this dialog. In this section you learn more about the difference between local clocks and NTP and how to set your system time.

No matter whether you live in New York, Budapest, Nakhodka, Ulan Bator, Bangkok, or Canberra, most of your Linux time computations are related to Coordinated Universal Time or UTC. If you run a dedicated Linux system, it is customary to keep the hardware clock set to UTC, but if you also boot another operating system such as Windows, you may need to set the hardware clock to local time. It really doesn't matter as far as Linux is concerned, except that there happen to be two different methods of keeping track of time zones internally in Linux, and if they don't agree, you can wind up with some odd time stamps on FAT filesystems, among other things. Listing 55 shows you how to use the `date` command to display the current date and time. The display is always in local time, even if your hardware clock keeps UTC time.

Listing 55. Displaying the current date and time

```
[root@lyrebird ~]# date;date -u
Mon Jul  9 22:40:01 EDT 2007
```

The `date` command supports a wide variety of possible output formats, some of which you already saw back in Listing 28. See the man page for `date` if you'd like to learn more about the various date formats.

If you need to set the date, you can do this by providing a date and time as an argument. The required format is historical and is somewhat odd even to Americans and truly odd to the rest of the world. You must specify at least month, day, hour, and minute in MMDDhhmm format, and you may also append a two- or four-digit year (CCYY or YY) and optionally a period (.) followed by a two-digit number of seconds. Listing 56 shows an example that alters the system date by a little over a minute.

Listing 56. Setting the system date and time

```
[root@lyrebird ~]# date; date 0709221407;date
Mon Jul  9 23:12:37 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
Mon Jul  9 22:14:00 EDT 2007
```

Set the BIOS clock to UTC time

Your Linux system, along with most other current operating systems, actually has two clocks. The first is the hardware clock, sometimes called the Real Time Clock, RTC, or BIOS clock, which is usually tied to an oscillating quartz crystal that is accurate to within a few seconds per day. It is subject to variations such as ambient temperature. The second is the internal software clock, which is driven by counting system interrupts. It is subject to variations caused by high system load and interrupt latency. Nevertheless, your system typically reads the hardware clock at startup and from then on uses the software clock. The `date` command that you just learned about sets the software clock, not the hardware clock.

If you use the Network Time Protocol (NTP), you may possibly set the hardware clock when you first install the system and never worry about it again. If not, this part of the tutorial will show you how to display and set the hardware clock time.

You can use the `hwclock` command to display the current value of the hardware clock. Listing 57 shows the current value of both the system and hardware clocks.

Listing 57. System and hardware clock values

```
[root@lyrebird ~]# date;hwclock
Mon Jul  9 22:16:11 EDT 2007
Mon 09 Jul 2007 11:14:49 PM EDT  -0.071616 seconds
```

Notice that the two values are different. You can synchronize the hardware clock

from the system clock using the `-w` or `--systohc` option of `hwclock`, and you can synchronize the system clock from the hardware clock using the `-s` or `--hctosys` option, as shown in Listing 58.

Listing 58. Setting the system clock from the hardware clock

```
[root@lyrebird ~]# date;hwclock;hwclock -s;date
Mon Jul  9 22:20:23 EDT 2007
Mon 09 Jul 2007 11:19:01 PM EDT  -0.414881 seconds
Mon Jul  9 23:19:02 EDT 2007
```

You may specify either the `--utc` or the `--localtime` option to have the system clock kept in UTC or local time. If no value is specified, the value is taken from the third line of /etc/adjtime.

The Linux kernel has a mode that copies the system time to the hardware clock every 11 minutes. This is off by default, but is turned on by NTP. Running anything that set the time the old fashioned way, such as `hwclock --hctosys`, turns it off, so it's a good idea to just let NTP do its work if you are using NTP. See the man page for `adjtimex` to find out how to check whether the clock is being updated every 11 minutes or not. You may need to install the adjtimex package as it is not always installed by default.

The `hwclock` command keeps track of changes made to the hardware clock in order to compensate for inaccuracies in the clock frequency. The necessary data points are kept in /etc/adjtime, which is an ASCII file. If you are not using the Network Time Protocol, you can use the `adjtimex` command to compensate for clock drift. Otherwise, the hardware clock will be adjusted approximately every 11 minutes by NTP. Besides showing whether your hardware clock is in local or UTC time, the first value in /etc/adjtime shows the amount of hardware clock drift per day (in seconds). Listing 59 shows two examples.

Listing 59. /etc/adjtime showing clock drift and local or UTC time.

```
[root@lyrebird ~]# cat /etc/adjtime
0.000990 1184019960 0.000000
1184019960
LOCAL
root@pinguino:~# cat /etc/adjtime
-0.003247 1182889954 0.000000
1182889954
LOCAL
```

Note that both these systems keep the hardware clock in local time, but the clock drifts are different — 0.000990 on lyrebird and -0.003247 on pinguino.

Configure your time zone

Your time zone is a measure of how far your local time differs from UTC. Information on available time zones that can be configured is kept in /usr/share/zoneinfo. Traditionally, /tec/localtime was a link to one of the time zone files in this directory tree, for example, /usr/share/zoneinfo/Eire or /usr/share/zoneinfo/Australia/Hobart. On modern systems it is much more likely to be a copy of the appropriate time zone data file since the /usr/share filesystem may not be mounted when the local time zone information is needed early in the boot process.

Similarly, another file, /etc/timezone was traditionally a link to /etc/default/init and was used to set the time zone environment variable TZ, and several locale-related environment variables. The file may or may not exist on your system. If it does, it may simply contain the name of the current time zone. You may also find time zone information in /etc/sysconfig/clock. Listing 60 shows these files from a Ubuntu 7.04 and a Fedora 7 system.

Listing 60. Time zone information in /etc

```
root@pinguino:~# cat /etc/timezone
America/New_York

[root@lyrebird ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of
/etc/localtime.
ZONE="America/New York"
UTC=false
ARC=false
```

Some systems such as Debian and Ubuntu have a `tzconfig` command to set the time zone. Others such as Fedora use `system-config-date` to set the time zone and to indicate whether the clock uses UTC or not. Listing 61 illustrates the use of the `tzconfig` command to display the current time zone.

Listing 61. Setting time zone with tzconfig

```
root@pinguino:~# tzconfig
Your current time zone is set to America/New_York
Do you want to change that? [n]:
Your time zone will not be changed
```

Configure the Network Time Protocol

The *Network Time Protocol* (*NTP*) is a protocol to synchronize computer clocks over a network. Synchronization is usually to UTC.

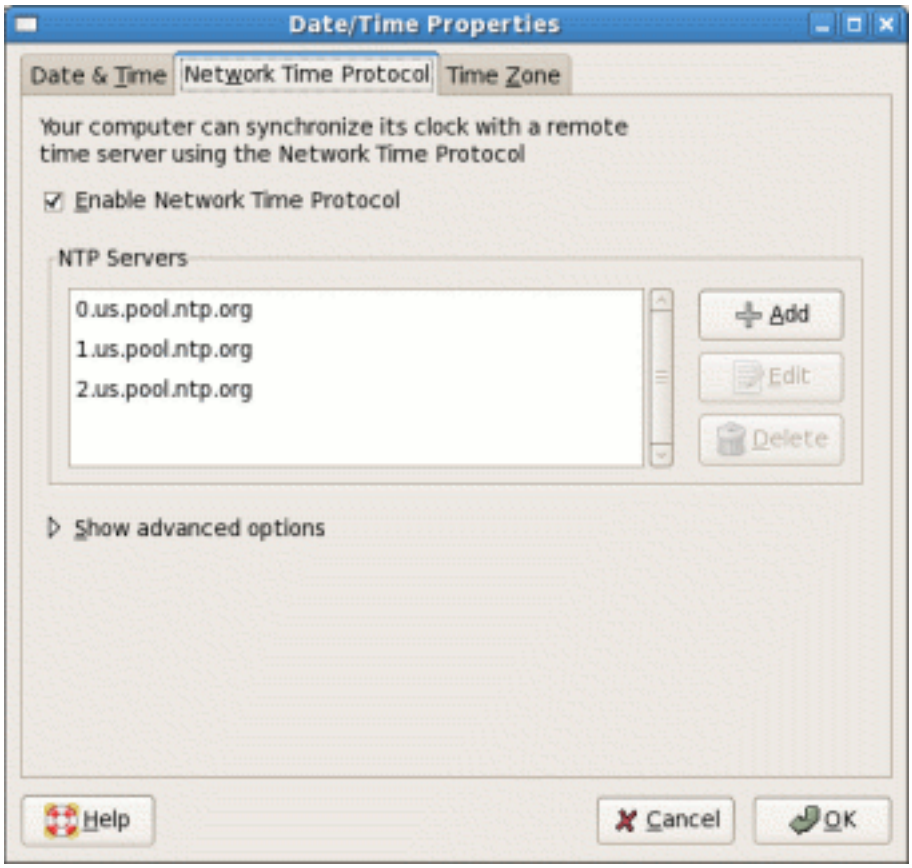
NTP version 3 is an Internet draft standard, formalized in RFC 1305. The current development version, NTP version 4 is a significant revision, which has not been formalized in an RFC. RFC 4330 describes Simple NTP (SNTP) version 4.

Time synchronization is accomplished by sending messages to *time servers*. The time returned is adjusted by an offset of half the round-trip delay. The accuracy of the time is therefore dependent on the network latency and the extent to which the latency is the same in both directions. The shorter the path to a time server, the more accurate the time is likely to be. See Resources for more detailed information than this simplistic description can provide.

There is a huge number of computers on the Internet, so time servers are organized into *strata*. A relatively small number of stratum 1 servers maintain very accurate time from a source such as an atomic clock. A larger number of stratum 2 servers get their time from stratum 1 servers and make it available to an even larger number of stratum 3 servers, and so on. To ease the load on time servers, a large number of volunteers donate time services through pool.ntp.org (see Resources for a link). Round robin DNS servers accomplish NTP load balancing by distributing NTP server requests among a pool of available servers.

If you use a graphical interface, you might be able to set your NTP time servers using a dialog similar to that in Figure 4. The fact that this system has enabled automatic time updates using NTP is why the dialog in Figure 3 did not allow the date and time to be changed.

Figure 4. Setting NTP servers



NTP configuration information is kept in /etc/ntp.conf, so you can also edit that file and then restart the ntpd daemon after you save it. Listing 62 shows an example /etc/ntp.conf file using the time servers from Figure 4.

Listing 62. Setting time zone with tzconfig

```
[root@lyrebird ~]# cat /etc/ntp.conf
# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface.  This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1

# Hosts on local network are less restricted.
#restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).

#broadcast 192.168.1.255 key 42         # broadcast server
#broadcastclient        # broadcast client
#broadcast 224.0.1.1 key 42             # multicast server
#multicastclient 224.0.1.1              # multicast client
#manycastserver 239.255.254.254         # manycast server
#manycastclient 239.255.254.254 key 42  # manycast client

# Undisciplined Local Clock. This is a fake driver intended for backup
# and when no outside source of synchronized time is available.
#server 127.127.1.0  # local clock
#fudge 127.127.1.0 stratum 10

# Drift file.  Put this in a directory which the daemon can write to.
# No symbolic links allowed, either, since the daemon updates the file
# by creating a temporary in the same directory and then rename()'ing
# it to the file.
driftfile /var/lib/ntp/drift

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys

# Specify the key identifiers which are trusted.
#trustedkey 4 8 42

# Specify the key identifier to use with the ntpdc utility.
#requestkey 8

# Specify the key identifier to use with the ntpq utility.
#controlkey 8
server 0.us.pool.ntp.org
restrict 0.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 1.us.pool.ntp.org
restrict 1.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
server 2.us.pool.ntp.org
restrict 2.us.pool.ntp.org mask 255.255.255.255 nomodify notrap noquery
```

If you are using the pool.ntp.org time servers, these may be anywhere in the world. You will usually get better time by restricting your servers as in this example where us.pool.ntp.org is used, resulting in only U.S. servers being chosen. See Resources

for more information on the ntp.pool.org project.

NTP commands

You can use the `ntpdate` command to set your system time from an NTP time server as shown in Listing 63.

Listing 63. Setting system time from an NTP server using ntpdate

```
[root@lyrebird ~]# ntpdate 0.us.pool.ntp.org
10 Jul 10:27:39 ntpdate[15308]: adjust time server 66.199.242.154 offset
-0.007271 sec
```

Because the servers operate in round robin mode, the next time you run this command you will probably see a different server. Listing 64 shows the first few DNS responses for 0.us.ntp.pool.org a few moments after the above `ntpdate` command was run.

Listing 64. Round robin NTP server pool

```
[root@lyrebird ~]# dig 0.pool.ntp.org +noall +answer | head -n 5
0.pool.ntp.org.  1062   IN      A       217.116.227.3
0.pool.ntp.org.  1062   IN      A       24.215.0.24
0.pool.ntp.org.  1062   IN      A       62.66.254.154
0.pool.ntp.org.  1062   IN      A       76.168.30.201
0.pool.ntp.org.  1062   IN      A       81.169.139.140
```

The `ntpdate` command is now deprecated as the same function can be done using `ntpq` with the `-q` option, as shown in Listing 65.

Listing 65. Setting system time using ntpd -q

```
[root@lyrebird ~]# ntpd -q
ntpd: time slew -0.014406s
```

Note that the `ntpd` command uses the time server information from /etc/ntp.conf, or a configuration file provided on the command line. See the man page for more information and for information about other options for `ntpd`. Be aware also that if the ntpd daemon is running, `ntpd -q` will quietly exit, leaving a failure message in /var/log/messages.

Another related command is the `ntpq` command, which allows you to query the NTP daemon. See the man page for more details.

This brings us to the end of this tutorial. We have covered a lot of material on system administration. Don't forget to rate this tutorial and give us your feedback.

Resources

Learn

- Review the entire LPI exam prep tutorial series on developerWorks to learn Linux fundamentals and prepare for system administrator certification.
- At the LPIC Program, find task lists, sample questions, and detailed objectives for the three levels of the Linux Professional Institute's Linux system administration certification.
- See the Partimage homepage for information on Partimage, a filesystem-aware partition dump and restore tool.
- "/etc: Host-specific system configuration" describes the Linux Standard Base (LSB) requirements for /etc.
- The Network Time Protocol Project produces a reference implementation of the NTP protocol, and implementation documentation.
- The Network Time Synchronization Project maintains an extensive array of documentation and background information, including briefing slides, on network time protocols.
- The pool.ntp.org project is a big virtual cluster of timeservers striving to provide reliable easy to use NTP service for millions of clients without putting a strain on the big popular timeservers.
- In "Basic tasks for new Linux developers" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- *LPI Linux Certification in a Nutshell, Second Edition* (O'Reilly, 2006) and *LPIC I Exam Cram 2: Linux Professional Institute Certification Exams 101 and 102 (Exam Cram 2)* (Que, 2004) are LPI references for readers who prefer book format.
- Find more tutorials for Linux developers in the developerWorks Linux zone.
- Stay current with developerWorks technical events and Webcasts.

Get products and technologies

- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the developerWorks community through our developer blogs, forums, podcasts, and community topics in our new developerWorks spaces.

About the author

Ian Shields
Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University. You can contact Ian at ishields@us.ibm.com.

Trademarks