

Chapter 3: The Linux Filesystem

1) Filesystems	3-2
2) Top Level Directories	3-3
3) Home Directories	3-5
4) Directory Commands.....	3-7
5) The /etc/passwd File.....	3-9
6) The /etc/group File	3-11
7) The newgrp command.....	3-12
8) The su command	3-13
9) File and Directory Permissions.....	3-14
10) chmod.....	3-18
11) umask.....	3-20

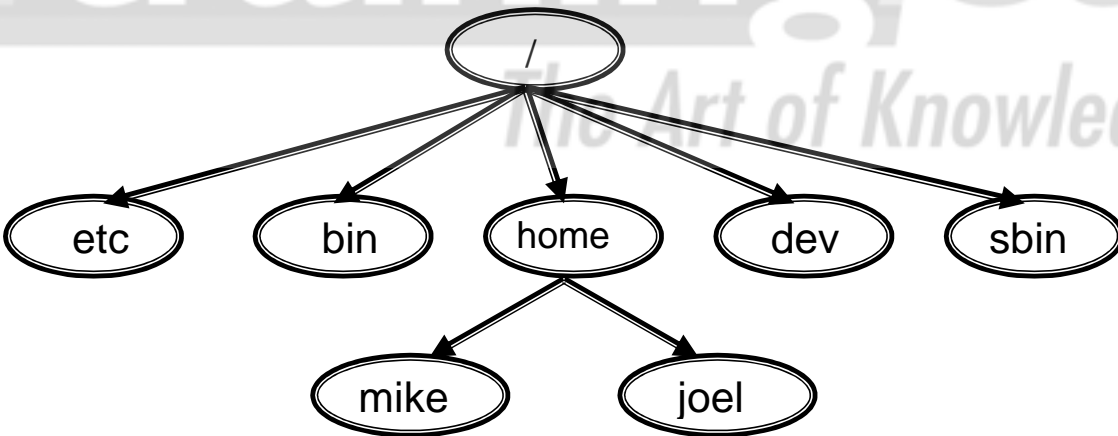
Evaluation
Copy

Filesystems

- In this section, we discuss the way in which Linux stores files.
- A loose definition of a file is "a collection of bytes" typically stored on the hard disk and associated with a name.
- There are many types of files:
 - ▶ ordinary file: text, program executable, image, ...
 - ▶ directory: a collection of files
 - ▶ device file: symbolic name for a physical device
 - ▶ others
- In the Linux operating system, each hard disk is usually sliced logically into partitions.
- Each partition can be formatted into a filesystem. This process is accomplished during the Linux installation. Further partitioning can also be accomplished after installation.
- The collection of all filesystems is referred to as "the" filesystem.
- Filesystems are studied in detail in a Linux System Administration course.

Top Level Directories

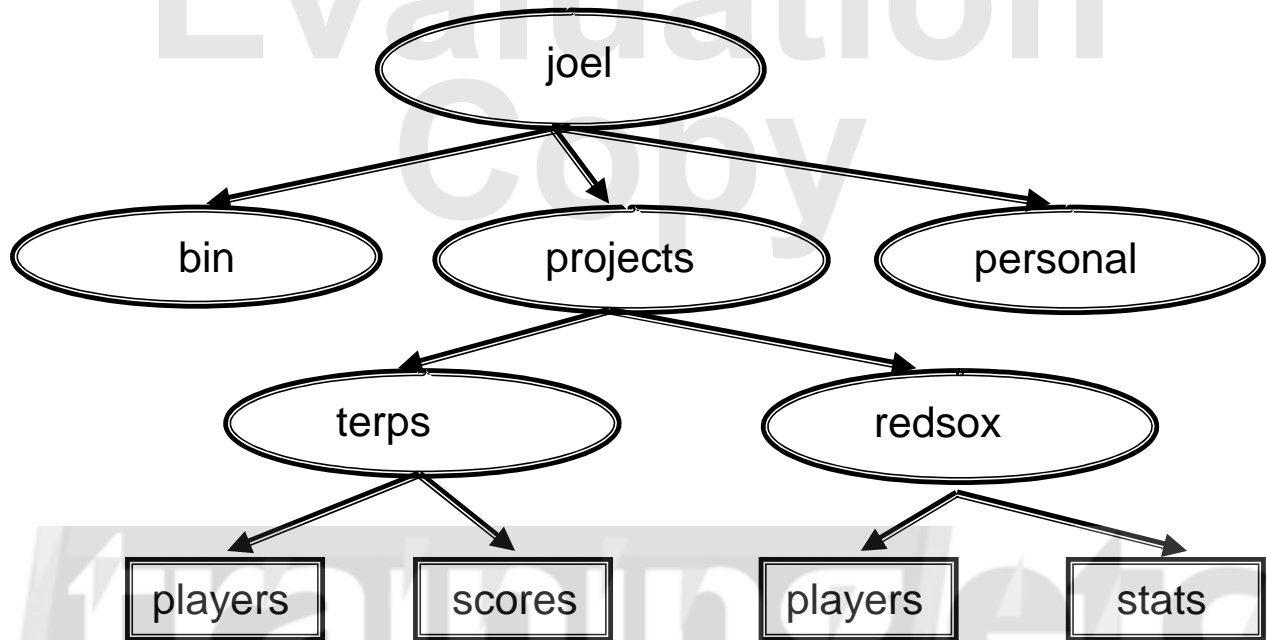
- One of the results of booting Linux is that all filesystems appear as one cohesive filesystem.
- After the boot process completes and the system is up and running, the entire filesystem can be viewed as an inverted tree.
- In this analogy, the top most directory is named / and pronounced "root."
- After installation, the root directory contains a well-known set of directories. Some of them are shown below.



Top Level Directories

- Each of the top level directories contains information essential for the operation of Linux.
 - ▶ etc: system administration configuration and data files
 - ▶ bin: binary executables (utilities)
 - ▶ dev: device special files
 - ▶ sbin: utilities for system administration
 - ▶ home: user's home directories
- The diagram is a representative sample of top level directories. There are other directories as well.
- The home directory is the place for the user's home directories. This is where you are "placed" when you login to the system.
 - ▶ mike would be placed in the directory `/home/mike`
 - ▶ joel would be placed in the directory `/home/joel`
- When users create a file, they would place it somewhere under their own directory. For example, perhaps joel would have created the following sub-tree under `/home/joel` found on the next page.

Home Directories



The Art of Knowledge.

Evaluation
Copy

Home Directories

- In the diagram on the previous page, joel has created three directories, bin, personal, and projects. The projects directory contains two subdirectories, terps and redsox.
- Each of these subdirectories has two ordinary files. terps has players and scores. redsox has players and stats.
- A complete path name is the name of a file or directory that includes all directory names back to the root directory.
- The complete path names for all files on the previous page are:

```
/home/joel
/home/joel/bin
/home/joel/projects
/home/joel/personal
/home/joel/projects/terps
/home/joel/projects/terps/players
/home/joel/projects/terps/scores
/home/joel/projects/redsox
/home/joel/projects/redsox/players
/home/joel/projects/redsox/stats
```

Directory Commands

- Suppose you are `joe1` and you have:
 - ▶ logged into the system
 - ▶ started a terminal window
- To verify the directory name where you have been placed upon login, you can execute the `pwd` command.
- Linux provides various permissions with regard to files and directories. Subject to these rules, you may navigate to most other directories on the system.
- The `cd` command allows you to change directories. In the session on the next page, you will see various uses of this command.
- You can also make one or more directories by using the `mkdir` command. You will see examples of this command in the session below.
- The `rmdir` command allows you to remove one or more directories, provided that they are empty.
- The `ls` command lists files and directories in the current directory. This command is more complex than meets the eye. Later we will demonstrate more of its functionality.

Directory Commands

- Here is a sample session. The shell ignores anything following the # sign.

```
$ pwd                # print working directory
/home/joel
$ mkdir bin personal projects temp
$ ls                 # list files and directories
bin    personal    projects    temp
$ cd projects       # change to projects
$ pwd
/home/joel/projects
$ mkdir terps redsox
$ ls
terps    redsox
$ cd ..            # change to parent directory
$ pwd
/home/joel
$ rmdir temp      # remove a directory
$ ls
bin    personal    projects
$
```

- The cd command has various "idioms."
 - ▶ cd - change to previous current directory
 - ▶ cd change to home directory
 - ▶ cd .. change to parent directory of current directory
 - ▶ cd bin change to bin relative to where you are
 - ▶ cd /bin change to /bin
 - ▶ cd ~mike change to mike's home directory, i.e. /home/mike

The `/etc/passwd` File

- In order to gain access to a Linux system, you must have an account.
- Accounts are created by the person(s) on your system who has been designated as the system administrator (SA). There may be more than one person with this title.
- As a result of account creation, there will be a line in the `/etc/passwd` file. This file is the de-facto user database and is usually spoken about as the "password" file.
- For security reasons, the real passwords are kept in the `/etc/shadow` file. Nevertheless, there is other important information in the password file.
- Each line of `/etc/passwd` contains seven fields, each of which describes information about a user.
 - ▶ Login name: 8 chars or less (joel, mike)
 - ▶ Password: x to indicate the use of `/etc/shadow`
 - ▶ User id: Unique integer so Linux can track you
 - ▶ Group id: Identifies you as belonging to this group
 - ▶ GECOS field: Not officially used by Linux
 - ▶ Home directory: Where you are placed on login
 - ▶ Program: Program to be run at login

The /etc/passwd File

- Here is an example of selected lines in a real /etc/passwd file.

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
nobody:x:99:99:Nobody:/:/sbin/nologin
user01:x:500:500:user01:/home/user01:/bin/bash
mike:x:527:527:./home/michael:/bin/bash
```

- The first line describes the `root` user, the one account on the system with almighty powers. Typically, only the system administrator(s) will know this password and use this account. In any case, this account should only be used when particular work cannot be accomplished through another account.

- Other accounts are pseudo users. They do not represent humans, but rather they represent special programs that can be executed simply by logging into the system.

```
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

- Still other accounts are simply there for ownership privileges and are otherwise unusable.

```
bin:x:1:1:bin:/bin:/sbin/nologin
```

- All of these "special accounts" have a UID less than 100. Real users will have a UID of at least 500.

The /etc/group File

- There is also a file named `/etc/group`. This file describes group information. Each line has four fields.
 - ▶ Group name: (joel, mike)
 - ▶ Password: Usually empty (Signified by an `x`)
 - ▶ Group id: Same as that in `/etc/passwd`
 - ▶ User list: Comma separated list of users in this group

- Each line in the "group" file describes a group. The most important feature here is the list of users that are part of this group.

- Use the `id` command to list your user identification and your group identification.

```
$ id
uid=527(mike) gid=527(mike) groups=527(mike)
$
```

- At any given instant you are part of one or more groups. If a particular group has you listed as a member, you can use the `newgrp` command to change your effective group id.

```
$ newgrp root
$ id
uid=527(mike) gid=0(root) groups=527(mike)
$
<<      user works for a while here      >>
$ exit
$ id
uid=527(mike) gid=527(mike) groups=527(mike)
$
```

The newgrp command

- When you use the `newgrp` command to change your effective group id, a new shell will be spawned for you. This shell terminates when you exit from it.
- You can see that there is a new shell running by using the `ps` command. In a later chapter we will discuss this command in more detail. Notice the following session.

```
$ ps
  PID  TTY          TIME CMD
 9848 pts/0    00:00:00 bash
 9979 pts/0    00:00:00 ps
$ newgrp root
$ ps
  PID  TTY          TIME CMD
 9848 pts/0    00:00:00 bash
 9899 pts/0    00:00:00 bash
 9979 pts/0    00:00:00 ps
$
<<      user works for a while here  >>
$ exit
$ ps
  PID  TTY          TIME CMD
 9848 pts/0    00:00:00 bash
 9979 pts/0    00:00:00 ps
$
```

- You can see that there is an *extra* `bash` running when you use the `newgrp` command.
- Becoming part of a new group affects the group ownership of any files and directories that you create.

The `su` command

- Sometimes you may wish to become a different user. That is you may wish to change both your effective user id (UID), and your effective group id (GID).
- You can always logout and then login as the new user but it is easier to use the `su` command.
- The `su` command switches you to another user. Here is an example of using the `su` command.

```
$ id
uid=527(mike) gid=527(mike) groups=527(mike)
$ su - root
Password: xxxxxx
uid=0(root) gid=0(root)
$
  << work as root for a while
$
$ exit
$ id
uid=527(mike) gid=527(mike) groups=527(mike)
$
```

- Notice that when you use the `su` command, both the UID and the GID will change.
- As was the case with `newgrp`, when you use `su`, a new bash is also launched.
- In the use of `su` above, the dash (`-`) was used. This causes the new user's startup files to be executed. If the dash is not used, then the startup sequence will not be executed.

File and Directory Permissions

- Most users think of files as having a name and having some content. While this is normally true, files also have other characteristics.
 - ▶ size
 - ▶ modification date
 - ▶ permissions
 - ▶ owner
 - ▶ group owner
 - ▶ ...
- This section discusses the permissions for files.
- From the point of view of a file, Linux maintains three categories of users:
 - ▶ the owner of a file
 - ▶ anybody in the group of the owner of the file
 - ▶ anybody else who can use this system that is not in the owner's group.
- For each of these classifications, there are potential permissions to read, write, and execute a file.
- This allows for a total of nine permissions (or modes as they are sometimes called).

File and Directory Permissions

- Permissions
 - ▶ Ordinary Files
 - Read: You can display it or print the file.
 - Write: You can change its contents. (not remove it)
 - Execute: You can execute the file as a command.
 -
 - ▶ Directories
 - Read: You can list files.
 - Write: You can create or remove files.
 - Execute: You can `cd` to this directory.
- Initially, a file or directory is given certain permissions by the program that created it.
 - ▶ compilers
 - ▶ `mkdir`
 - ▶ `bash`
 - ▶ editors
- Beyond the initial permissions established for a file or a directory, only the owner (or the root user) can change the permissions.
- To actually see the permissions for a file, you can use the `stat` command or the `ls` command with the `-l` option.
- Of course we will also need to create a file or two. You can use the `touch` command to create an empty file.

File and Directory Permissions

- The session below creates a few files and displays their permissions as well as other information about them.

```
$ touch test data
$ ls
test  data
$ ls -l
total 8
-rw-r--r--  1 joel   joel   0 Mar 30 08:27 data
-rw-r--r--  1 joel   joel   0 Mar 30 08:27 test
$
```

- Each line above (except the first line) gives information about a file. The first ten characters on each line gives the following information:

- ▶ First character:

- d directory
- - ordinary file

- ▶ Characters 2-10

- 2-4 owner: read, write, execute permissions
- 5-7 group: read, write, execute permissions
- 8-10 others: read, write, execute permissions

- For the files `test` and `data`, `joel` has read and write permissions. Users in `joel`'s group, and all other users not in `joel`'s group, will have read permissions.

File and Directory Permissions

- In the session below, several directories are created. To see the permissions for these directories, use the `-l` option of the `ls` command.

```
$ mkdir bin project
$ ls
bin          data        project     test
$ ls -l
total 8
drwxr-xr-x  2 joel   joel  4096 Mar 30 08:48 bin
-rw-rw-rw-  1 joel   joel    0 Mar 30 08:27 data
drwxr-xr-x  2 joel   joel  4096 Mar 30 08:48 project
-rw-rw-rw-  1 joel   joel    0 Mar 30 08:27 test
$
```

- Note that the two newly created directories have all permissions turned on for the owner. This means that the owner of these directories can list files in either directory, create and remove files, and `cd` to these directories. The users in the owners group and everybody else have the ability to list files and `cd` to the directories.
- It is not always desirable for non-owners of a directory to be able to see the contents of a directory, or to be able to create and destroy files in a directory they do not own.
- As the owner of this directory or as the owner of any file, you can change the modes of a file with the `chmod` command.

chmod

- The `chmod` command takes at least two arguments. The first argument is the new mode for the file. The remaining arguments are a list of files whose mode will change.

```
$ chmod 644 test
$ chmod uo-r data
```

- As you can see, there are two ways in which to alter the mode of a file. Either octal numbers or symbolic letters may be used.

- Using octal digits:

- ▶ Consider the three permissions for each of the three classes of users: owner, group, and other.
- ▶ For each of these classes, determine the permissions you wish them to have.
- ▶ For example, let's say we want all classes to have read permissions but only the owner to have write permissions.

Owner	Group	Other
r w x	r w x	r w x
Y Y N	Y N N	Y N N

- ▶ Now count 4 for read, 2 for write, and 1 for execute, and then total each group.

Owner	Group	Other
r w x	r w x	r w x
Y Y N	Y N N	Y N N
4 2 0	4 0 0	4 0 0

```
$ chmod 644 test
```

chmod

- Using symbolic letters:

- ▶ Each group has a symbolic letter
 - o other (other users not in the file's group)
 - g group
 - u owner
 - a all

- ▶ Each permission has a symbolic letter

- r read
- w write
- x execute

- ▶ Each action has a symbolic letter

- - take away
- + add
- = assign

- Examples:

- ▶ All classes get read and write access.

```
$ chmod a=rw data
```

- ▶ Take away write permissions from group and other.

```
$ chmod og-w data
```

- There are three additional permissions, `setuid`, `setgid`, and the sticky bit. They will be discussed in a later chapter.

umask

- Linux also allows an additional feature that each user can employ. It is called the `umask` (user mask).
- When you use the user mask, you are specifying which permissions that you want to subtract from those permissions used by the creation program.
- Without the `umask`, we have seen the defaults with which files are created.
- To see the current `umask` or to set it, proceed as in the sample session shown here.

```
$ umask
0022
$ touch newfile
$ ls -l newfile
-rw-r--r--  1 joel joel 0 Mar 30 13:30 newfile
$ umask 077
$ touch myfile
$ ls -l myfile
-rw-----  1 joel joel 0 Mar 30 13:35 myfile
$
```

Exercises

1. How many directories are there under the / directory? Count only first-level directories. Do you have to `cd` to the / directory to answer this question?

2. Observe the diagram on page five. Which directory would you be in at the end of the session below?

```
$ pwd
/home/joel/projects
$ cd terps
$ cd ../../..
```

3. Write two different `cd` commands that will take you from /home/joel/projects to /home/joel.

4. Write three different `cd` commands that will take you to your home directory.

5. What do each of the following commands do?

```
$ echo ~
$ echo ~student
$ cd ~
$ cd -
$ cd ~student
```

6. Use the `chmod` command to change the permissions of the `testfile` file so that when you perform an `ls -l` on `testfile`, the permissions look like what is shown below.

```
$ ls -l testfile
-r---w---x .....
$
```

Exercises

- Now change the mode of `testfile` so that the permissions are `read/write` for owner, `read/execute` for group, and `read` for everybody else.
- Create the following directory structure underneath your home directory.

```
mon
mon/progs
mon/progs/c
mon/progs/c++
mon/scripts
mon/scripts/bash
mon/scripts/csh
tue
tue/progs
tue/progs/c
tue/progs/c++
tue/scripts
tue/scripts/bash
tue/scripts/csh
```

- There is a Linux command named `find` that locates where files are in a directory structure. Execute the following `find` command to verify the directory structure that you have just built.

```
$ find mon tue
```

- How many users have user id's ≥ 500
- What is your user id.
- What is the user id of the account named `root`.
- How many other users are in your group?

Exercises

14. This exercise has many parts:
15. Assure that your home directory has permissions that allow only you and users in your group to create and remove files. If that is not the case, change the permissions on your home directory so that it is.
16. Use the `su` command to switch your effective user id to `michael` and then change to the `student` home directory as shown below. This assumes that there is a `michael` account.

```
$ su - michael
Password:
$ cd ~student
```

Use the `id` command to see your UID and GID

Now create a file there called `myfile`.

```
$ date > myfile      # create myfile
```

Note the group and owner of this file when you execute the `ls -l` command.

Remember that the `su` command creates a new shell. Using the `exit` command, exit from that shell now and then execute the `id` command again.

Exercises

Now use the `su` command to become user `erin` and change to the student home directory. This assumes that the account `erin` exists.

```
$ su - erin
Password:
$ cd ~student
```

Try to create a file there as before.

Why is it that you cannot create this file as `erin` but you could as `michael`.

Exit from this shell.

Change the permissions of your home directory to 755.

Evaluation
Copy

Exercises

17. This exercise has many parts.

- a) Execute the `id` command so that you will see your `UID` and `GID`. You will also see any other groups of which you may be a member.
- b) if you are not a member of any other groups, use the `su` command to switch to a user who is in more than one group.
- c) Now execute the command shown below to change to a new group.

```
$ newgrp susan
```

- d) execute the `id` command again to see the affect of this change.
- e) execute the `ps` command to see that an additional shell has been launched.
- f) Use the following command to create a new file.

```
$ date > datefile
```

- g) Use the `ls -l` command to determine the group ownership of this new file.
- h) use the `exit` command to end the `newgrp` session. Now execute the `id` command to see your `UID` and `GID` once again. Finally execute the `ps` command to see that the shell spawned with `newgrp` has exited.

This Page Intentionally Left Blank

Evaluation
Copy



Evaluation
Copy