SISTEM OPERASI LANJUTAN

#3 UNIX/Linux File System & File Security

Objectives

- Discuss UNIX/Linux file systems
- Explain partitions and inodes
- Understand the elements of the root hierarchy
- Use the *mount* command
- Explain and use paths, pathnames, and prompts

Objectives (continued)

- Navigate the file system
- Create and remove directories
- Copy and delete files
- Configure file permissions

Understanding UNIX/Linux File Systems

- **File**: basic component for data storage
- **File system**: UNIX/Linux system's way of organizing files on storage devices
  - **Physical file system**: section of the hard disk that has been formatted to hold files
- UNIX/Linux consist of multiple file systems that form **virtual storage** space for multiple users
- UNIX/Linux systems support many file systems
  - Examples: UNIX file system (ufs), extended file system (ext or ext fs)

Understanding UNIX/Linux File Systems (continued)

- **ufs**: original native UNIX file system
  - Expandable, supports large amounts of storage, provides excellent security, reliable
  - Supports **journaling**
  - Supports **hot fixes**
- In Linux, the native file system is **ext**
  - Installed by default
  - Modeled after ufs
    - First version contained some bugs
  - Newer versions of Linux use ext2, ext3, or ext4
    - ext4 enables the use of **extents**

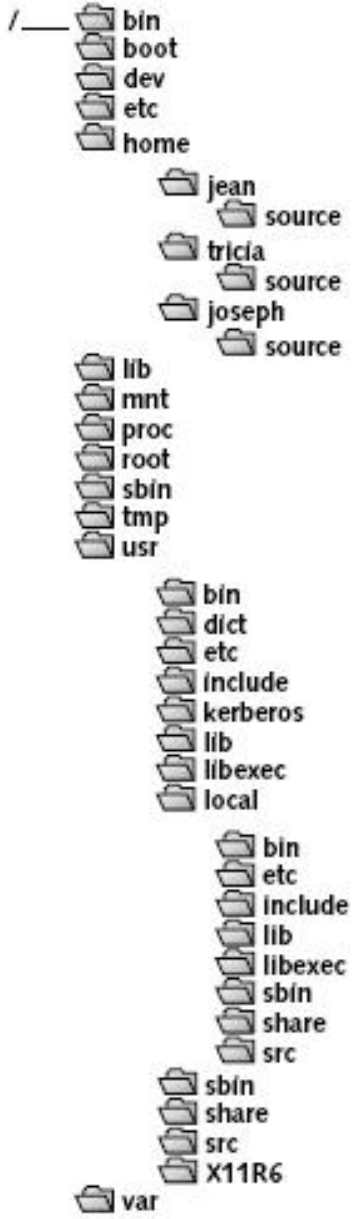

Figure 2-1 Typical UNIX/Linux hierarchical structure

Table 2-1 Typical file systems supported by UNIX/Linux

File System	Description
Extended file system (ext or ext fs) and the newer versions: second extended file system (ext2 or ext2 fs), third extended file system (ext3 or ext3 fs), and fourth extended file system (ext4 or ext4 fs)	Comes with Linux by default (compatible with Linux and FreeBSD); ext3 offers journaling, which is important for reliability and recovery when a system goes down unexpectedly; ext4 adds larger volume sizes plus extents
High-performance file system (HPFS)	Developed for use with the OS/2 operating system
International Organization for Standardization (ISO) Standard Operating System 9660 (iso9660 in Linux, hsfs in Solaris, cd9660 in FreeBSD)	Developed for CD and DVD use; does not support long file names
Journaled File System (JFS)	Modeled after IBM's JFS; offers mature journaling features, fast performance for processing larger files, dynamic inode allocation for better use of free space, and specialized approaches for organizing either small or large directory structures
msdos	Offers compatibility with FAT12 and FAT16 (does not support long file names); typically installed to enable UNIX to read floppy disks made in MS-DOS or Windows
Network file system (NFS)	Developed by Sun Microsystems for UNIX systems to support network access and sharing of files (such as uploading and downloading); supported on virtually all UNIX/Linux versions as well as many other operating systems
NT file system (NTFS)	Used by Windows NT, Windows 2000, Windows XP, Windows Vista, and Windows Server systems
Proc file system	Presents information about the kernel status and the use of memory (not truly a physical file system, but a logical file system)

Table 2-1 Typical file systems supported by UNIX/Linux (continued)

File System	Description
ReiserFS	Developed by Hans Reiser and similar to ext3 and ext4, with journaling capabilities; designed to be faster than ext3 and ext4 (up to 15 times) for handling small files; intended to encourage programmers to create efficient code through use of smaller files
Swap file system	File system for the swap space—that is, disk space used exclusively to store spillover information from memory when memory is full (called virtual memory) and used by virtually all UNIX/Linux systems; on newer UNIX/Linux systems, the swap file system is encrypted for improved security
Universal Disk Format (UDF)	Developed for CD and DVD use and broadly replacing iso9660. UDF read capability is supported in Windows, UNIX/Linux, and Mac OS systems prior to 2006; read/write capability is supported in Windows Vista, UNIX/Linux versions after 2005, and Mac OS Tiger and the newer Leopard.
uMS/DOS	Compatible with extended FAT16 as used by Windows NT, 2000, XP, Vista, and Server, but also supports security permissions, file ownership, and long file names
UNIX file system (ufs; also called the Berkeley Fast File System)	Original file system for UNIX; compatible with virtually all UNIX systems and most Linux systems
vfat	Compatible with FAT32 and supports long file names
XFS	Silicon Graphics' file system for the Irix version of UNIX; offers many types of journaling features and is targeted for use with large disk farms (multiple disk storage devices available through high-speed connections)

Table 2-2 Comparison of typical file systems supported by UNIX/Linux

Feature	FAT	NTFS	ext4	ufs
Total volume or partition size	2 GB to 2 TB	2 TB	1 exabyte in Linux depending on the kernel version*	1 TB in Linux; 4 GB to 2 TB in UNIX depending on the version
Maximum file size	2 GB for FAT16; 4 GB for FAT32	Potentially 16 TB, but limited by the volume size (up to 2 TB)	16 GB to 2 TB in Linux depending on the kernel version*	2 GB in Linux; 2 GB to 16 TB in UNIX depending on the version
Security	Limited security based on attributes and shares	Extensive security through permissions, groups, and auditing options	Extensive security through permissions and groups	Extensive security through permissions and groups
Reliability through file activity tracking or journaling	None	Journaling	Journaling	Journaling
POSIX support	None (FAT16); limited (FAT32)	Yes	Yes	Yes
Reliability through hot fix capability	Limited	Supported	Supported	Supported
Support for extents	No	Yes, when preallocated via a program	Yes, when enabled	No

*These maximums are limited by the kernel version and are based on Linux kernel version 2.6.19.

Understanding the Standard Tree Structure

- The treelike structure for UNIX/Linux file systems starts at the root file system level
  - Root is denoted by /
    - Slash represents the **root file system directory**
- **Directory**: special kind of file that can contain other files and directories
  - May have **subdirectories**
    - Subdirectory is considered **child** of **parent** directory

Using UNIX/Linux Partitions

- **Partition**: section of disk that holds a file system
  - UNIX/Linux partitions identified with names
    - Examples: hda1, sda1
      - First two letters tell Linux the device type
      - Third letter indicates if disk is the primary or secondary disk
      - Partitions on a disk are numbered starting with 1
- **Peripherals** connect through electronic interfaces
  - Examples of hard disk interfaces: **IDE**, **SCSI**, **EIDE**

```
Disk/dev/hda: 128 heads, 63 sectors, 767 cylinders
Units = cylinders of 8064 * 512 bytes

Device     Boot   Begin     Start      End      Blocks   Id  System
/dev/hda1    *        1         1      242      975712+   6  DOS 32-bit >=32M
/dev/hda2             243       243      767     2116899    5  Extended
/dev/hda3             243       243      275      127024+  83  Linux native
/dev/hda6             276       276      750     1028224+  83  Linux native
/dev/hda7             751       751      767       68512+  82  Linux swap

Command (m for help):  _
```

This partition table is from a Linux system with an IDE drive

```
Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes

Device     Boot   Begin     Start      End      Blocks   Id  System
/dev/sda1             1         1       64      514048+  83  Linux native
/dev/sda2            65        65     1106     8369865    5  Extended
/dev/sda5            65        65     1084     8193118+  83  Linux native
/dev/sda6          1085      1085     1100      128488+  82  Linux swap

Command (m for help):  _
```

This partition table is from a Linux system with a SCSI drive

Figure 2-2 Sample Linux partition tables

Setting Up Hard Disk Partitions

- Partition to organize space to contain file systems
- Some UNIX/Linux vendors recommend that:
  - Root partition holds the root file system directory
  - **Swap partition** acts like an extension of memory
    - General rule: same size as RAM
    - A swap partition enables **virtual memory**
  - **/boot partition** to store OS kernel files
- Other partitions:
  - **/usr** (for **utilities**), **/home**, **/var**
- Mount partition to become part of file system

Using Inodes

- **Information nodes**, or **inodes**
  - Each directory/file has an inode and is identified by an inode number
    - Inode 0 contains the root of the directory structure (/)
      - Jumping-off point for all other inodes
  - Contains file/directory name, general information, pointer to the directory/file on a disk partition
- **Superblock** contains information about the layout of blocks on a specific partition

Exploring the Root Hierarchy

- The root (/) file system is mounted by the kernel when the system starts
  - To **mount** a file system is to connect it to the directory tree structure
    - System administrator uses *mount* command
  - Root file system contains all essential programs for file system repair
    - Restoring from a backup
    - Starting the system
    - Initializing all devices and operating resources
    - Information for mounting other file systems

The /bin Directory

- Contains **binaries**, or **executables**
  - Programs needed to start the system and perform other essential system tasks
- Holds many programs that all users need to work with UNIX/Linux

The /boot Directory

- Normally contains:
  - Files needed by the bootstrap loader
    - The **bootstrap loader** is the utility that starts the OS
  - Kernel (OS) images

The /dev Directory

- Files in /dev reference system devices
- Devices are managed through **device special files**
  - Contain information about I/O devices that are used by OS kernel when a device is accessed
  - Two types:
    - **Block special files**
      - Example: for CD/DVD drives
    - **Character special files**
      - Example: for printers
  - To see the list of device files: *ls -l /dev*
  - *null* is a “black hole”

Table 2-3 UNIX/Linux device special files

File	Description
/dev/console	For the console components, such as the monitor and keyboard attached to the computer (/dev/tty0 is also used at the same time on many systems)
/dev/fd*n*	For floppy disk drives, where *n* is the number of the drive, such as fd0 for the first floppy disk drive
/dev/hd*xn*	For IDE and EIDE hard drives, where *x* represents the disk and the *n* represents the partition number, such as hda1 for the first disk and partition
/dev/modem	For a modem, a symbolic link to the device special file (typically linked to /dev/ttys1), where a symbolic link enables one file or directory to point to another (in later versions of Fedora/Red Hat Enterprise Linux, the modem file may be in /usr/share/applications, and in SUSE this file may be under /usr/share/applications/YaST2 because it is managed using the YaST management tool)
/dev/mouse	For a mouse or other pointing device, a symbolic link to the device special file (typically linked to /dev/ttys0)—in Fedora/Red Hat Enterprise Linux, the mouse file may be under /usr/share/applications, and in SUSE it may be under /opt/gnome/share/applications
/dev/sd*xn*	For a hard drive connected to a SCSI interface, where *x* represents the disk and the *n* represents the partition, such as sda1 for the first SCSI drive and first partition on that drive
/dev/st*n*	For a SCSI tape drive, where *n* represents the number of the drive, such as st0 for the first tape drive
/dev/tty*n*	For serial terminals connected to the computer
/dev/ttys*n*	For a serial device connected to the computer, such as ttys0 for the mouse

Some block devices in /dev

```
brw-rw-r--    1 root     floppy     2,     0 May  5 2008 fd0
brw-rw----    1 root     disk       3,     0 May  5 2008 hda
brw-rw----    1 root     disk       3,     1 May  5 2008 hda1
brw-rw----    1 root     disk       3,    64 May  5 2008 hdb
brw-rw----    1 root     disk       3,    65 May  5 2008 hdb1
brw-r-----    1 root     disk       1,     1 May  5 2008 ram
brw-rw----    1 root     disk      11,     0 May  5 2008 scd0
brw-rw----    1 root     disk      11,     1 May  5 2008 scd1
brw-rw----    1 root     disk       8,     0 May  5 2008 sda
brw-rw----    1 root     disk       8,     1 May  5 2008 sda1
brw-rw----    1 root     disk       8,    16 May  5 2008 sdb
brw-rw----    1 root     disk       8,    17 May  5 2008 sdb1
```

File type	Meaning
-	Normal
d	Subdirectory
b	Block device
c	Character device

Some character devices in /dev

```
crw-------    1 root     root       4,     0 Jan  4 01:07 console
crw-rw----    1 root     uucp       5,    64 Jan  4 01:07 cua0
crw-rw----    1 root     uucp       5,    65 May  5 2008 cua1
crw-rw----    1 root     uucp       5,    66 May  5 2008 cua2
crw-rw----    1 root     uucp       5,    67 May  5 2008 cua3
crw-rw-rw-    1 root     root      44,     0 May  5 2008 cui0
crw-rw----    1 root     daemon     6,     0 May  5 2008 lp0
crw-rw----    1 root     daemon     6,     1 May  5 2008 lp1
crw-r-----    1 root     kmem       1,     1 May  5 2008 mem
crw-rw-rw-    1 root     root       1,     3 May  5 2008 null
crw-rw-rw-    1 root     tty        2,   176 May  5 2008 ptya0
crw-rw-rw-    1 root     tty        2,   177 May  5 2008 ptya1
crw-rw-rw-    1 root     root       5,     0 May  5 2008 tty
crw-------    1 jdent    jdent      4,     0 May  5 2008 tty0
crw-r--r--    1 root     root       4,    65 Jan  4 18:29 ttyS1
```

Figure 2-3 Device files in /dev

The /etc Directory

- Contains configuration files that the system uses when the computer starts
  - *fstab*
  - *group*
  - *inittab*
  - *login.defs*
  - *motd*
  - *passwd*
  - *printcap* and *termcap*
  - *profile*, *bashrc* and *rc*

The /home Directory

- Often located on the /home partition
- Used to offer disk space for users, such as on a system that has multiple user accounts
  - Examples:
    - /home/jean
    - /home/tricia
    - /home/joseph

The /lib Directory

- /lib houses:
  - Kernel modules
  - Security information
  - **Shared library images**
    - Used by programmers to share code rather than creating copies in their programs
- Many files in this directory are symbolic links to other library files
  - **Symbolic link:** name, file name, or directory name that contains a pointer to a file/directory in the same directory or in another directory on your system

The /mnt Directory

- Mount points for temporary mounts by the system administrator reside in /mnt
  - A temporary mount is used to mount a removable storage medium
    - Example: CD/DVD or USB/flash storage
- /mnt is often divided into subdirectories to clearly specify device types
  - Example: /mnt/cdrom

The /media Directory

- In newer distributions of UNIX/Linux, mount points for removable storage are in /media
  - Relatively new recommendation of the Filesystem Hierarchy Standard (FHS)
- Modern Linux distributions include both /mnt and /media directories
  - Users and programmers are often encouraged to use /media

The /proc Directory

- /proc occupies no space on the disk
  - **Virtual file system** allocated in memory only
- Files in /proc refer to various processes running on the system as well as details about the OS kernel

The /root Directory

- Home directory for the root user
  - The system administrator

The /sbin Directory

- Reserved for the system administrator
- Stores:
  - Programs that start the system
  - Programs needed for file system repair
  - Essential network programs

The /tmp Directory

- Many programs need a temporary place to store data during processing cycles
  - The traditional location for these files is /tmp

The /usr Directory

- Houses software offered to users
  - Software might be:
    - Accounting programs
    - Manufacturing programs
    - Programs for research applications
    - Office software
- Frequently located on the /usr partition

The /var Directory

- Located on the /var partition
- Holds subdirectories that often change in size
  - These subdirectories contain files such as error logs and other system performance logs
  - Common subdirectories are:
    - /var/spool/mail for incoming mail
    - /var/spool/lpd for temporarily holding print files

Using the mount Command

- Use *mount* to connect the file system partitions to the directory tree when the system starts

Syntax **mount** [-option] [*device-name mount-point*]

Dissection

- Use the *-t* option to specify a file system to mount.
- *device-name* identifies the device to mount.
- *mount-point* identifies the directory in which you want to mount the file system.

- Example:

```
mount -t iso9660 /dev/cdrom /media/cdrom
```

- Use *umount* before removing the storage media

```
umount /media/cdrom
```

Using Paths, Pathnames, and Prompts

- Files are stored in directories in the file system, starting from the root file system directory
- To specify a file or directory, use its **pathname**
  - Follows the branches of the file system to the desired file
- A forward slash (/) separates each directory name
  - Example: /home/jean/source/phones.502

Using and Configuring Your Command-Line Prompt

~ is shorthand for the home directory

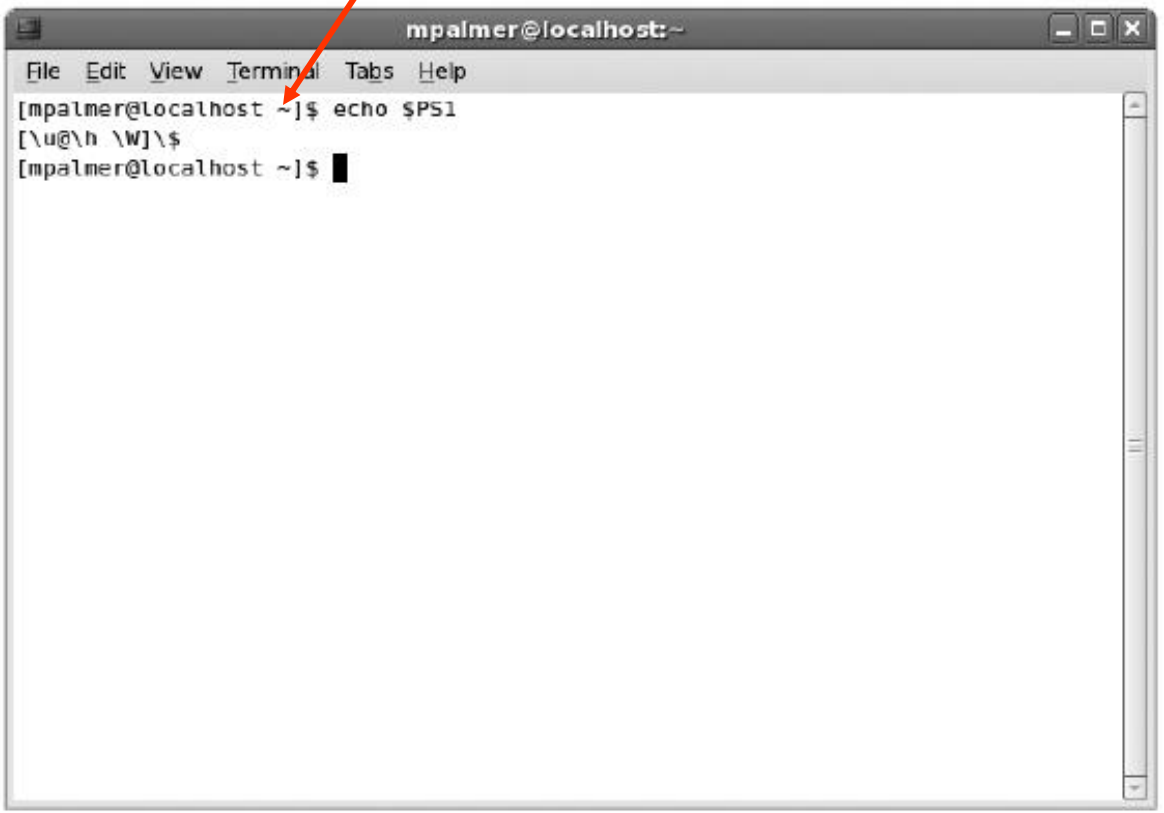

Figure 2-4 Viewing the contents of the PS1 variable

Table 2-4 Formatting characters for configuring a Bash shell prompt

Formatting Character	Purpose
\a	Sounds an alarm
\d	Displays the date
\e	Uses an escape character
\h	Displays the host name
\j	Shows the number of background jobs
\n	Displays a new line
nnn	Displays the ASCII character that corresponds to the octal number *nnn*
\r	Places a carriage return in the prompt
\s	Displays the shell name
\t	Displays the time
\u	Displays the username
\v	Displays the Bash version and release number
\w	Displays the path of the working directory
\A	Displays the time in 24-hour format
\D(format)	Displays the time in a specific format
\H	Has the same effect as \h
\T	Displays the time in 12-hour format
\V	Displays the Bash version, release number, and patch level
\W	Displays the name of the working directory without any other path information
\!	Displays the number of the current command in the command history
\#	Displays the number of the command in the current session
\$	Displays a # if root is the user, otherwise displays a $
\@	Displays the time in 12-hour format
$PWD	Displays the path of the current working directory
\[	Marks the beginning of a sequence of nonprinting characters, such as a control sequence
\]	Marks the end of a sequence of nonprinting characters
\\	Displays a \ character

The pwd Command

- *pwd* prints the working directory

Syntax **pwd**

Dissection

- Use *pwd* to determine your current working directory.
- Typically, there are no options with this command.

- Useful for regular users, system administrators, and in scripts

Navigating the File System

- *cd* stands for change directory

Syntax **cd** [*directory*]

Dissection

- *directory* is the name of the directory to which you want to change. The directory name is expressed as a path to the destination, with slashes (/) separating subdirectory names.

- Provide an absolute or relative path to the directory
  - **Absolute path**: begins at the root level and lists all subdirectories to the destination file
    - Example: `cd /home/jean/source`
  - **Relative path**: takes a shorter journey
    - Example: `cd source or cd`

Using Dot and Dot Dot Addressing Techniques

- A single dot character means the current working directory
- Dot dot means the parent directory
- These addressing mechanisms are useful when navigating the file system
  - Example: `cd ../tricia/source`

Listing Directory Contents

- Use the *ls* (list) command to display a directory's contents, including files and other directories

Syntax **ls** [-option] [*directory or filename*]

Dissection

- Common arguments include a directory name (including the path to the directory) or a file name.
- Useful options include:
  - *-l* to view detailed information about files and directories
  - *-S* to sort by size of the file or directory
  - *-X* to sort by extension
  - *-r* to sort in reverse order
  - *-t* to sort by the time when the file or directory was last modified
  - *-a* to show hidden files ← Appear with a dot at the beginning
  - *-i* to view the inode value associated with a directory or file

Listing Directory Contents (continued)

Group (root)

Size (4096 bytes)

Owner (root)

Date and time of last modification

Number of links (2)

File or directory name (bin)

File type and access permissions: **drwxr-xr-x**

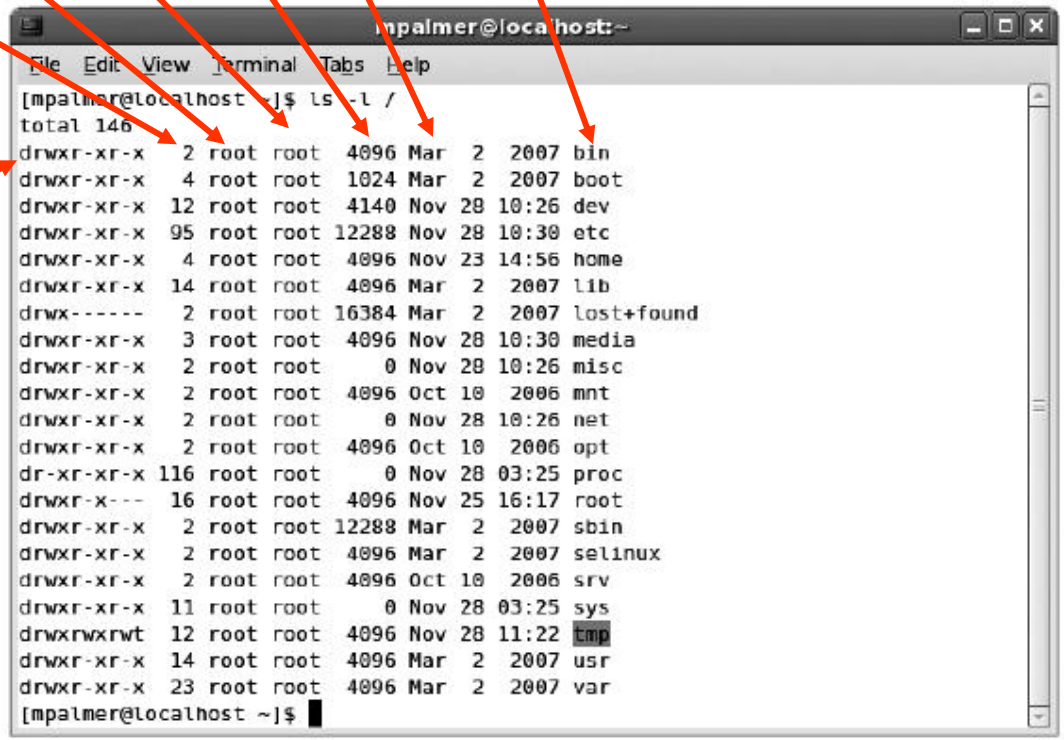

Figure 2-5 Using *ls -l* to view the root file system directory contents

Using Wildcards

- **Wildcard:** special character that can stand for any other character or a group of characters
  - * represents any group of characters in a file name
    - Example: *ls *.txt*

```
instructions.txt minutes.txt
```

  - ? takes the place of only a single character
    - Example: *ls list?*

```
list1 list2
```

Creating and Removing Directories

- *mkdir* is used to create a new directory

Syntax **mkdir** [-option] *directory*

Dissection

- The argument used with *mkdir* is a new directory name.
- There are only a few options used with *mkdir*. One option is to use *-v* to display a message that verifies the directory has been made.

- Delete empty directories using *rmdir*

Syntax **rmdir** [-option] *directory*

Dissection

- The argument used with *rmdir* is a directory.
- As is true for *mkdir*, *rmdir* has only a few options. Consider using the *-v* option to display a message that verifies the directory has been removed.

  - Use *rm -r* to delete a directory that is not empty

Copying and Deleting Files

- Use *cp* to copy files and *rm* to delete them

Syntax **cp** [-option] *source destination*

Dissection

- The argument consists of the source and destination directories and files, such as *cp /home/myaccount/myfile /home/youraccount*.
- Common options include:
  - *-b* makes a backup of the destination file if the copy will overwrite a file
  - *-i* provides a warning when you are about to overwrite a file
  - *-u* specifies to only overwrite if the file you are copying is newer than the one you are overwriting

Syntax **rm** [-option] *filename*

Dissection

- The argument consists of the name of the file to delete.
- The *-i* option causes the operating system to prompt to make certain you want to delete the file before it is actually deleted.

Configuring File Permissions for Security

- Users can set **permissions** for files/directories they own so as to establish security
  - System administrators also set permissions to protect system and shared files
- Permissions manage who can read, write, or execute files
- Original file owner of a file is the account that created it
  - File ownership can be transferred to another account

Configuring File Permissions for Security (continued)

File type	Meaning
-	Normal file
d	Subdirectory
l	Symbolic link
b	Block device file
c	Character device file

```
Excerpt from ls -l /etc

drwxr-xr-x   16 root    root         4096   Jan  17    9:29   X11
-rw-r--r--    1 root    root           46   Jan  15   19:11   adjtime
drwxr-xr-x    1 root    root         1024   Feb  27    2007   cron.daily
```

```
Excerpt from ls -l /home/jean/source

rw-rw-r--     1 jean    jean          387   Dec  12   23:11   phones.502
```

Figure 2-6 File types described in directory listings

Configuring File Permissions for Security (continued)

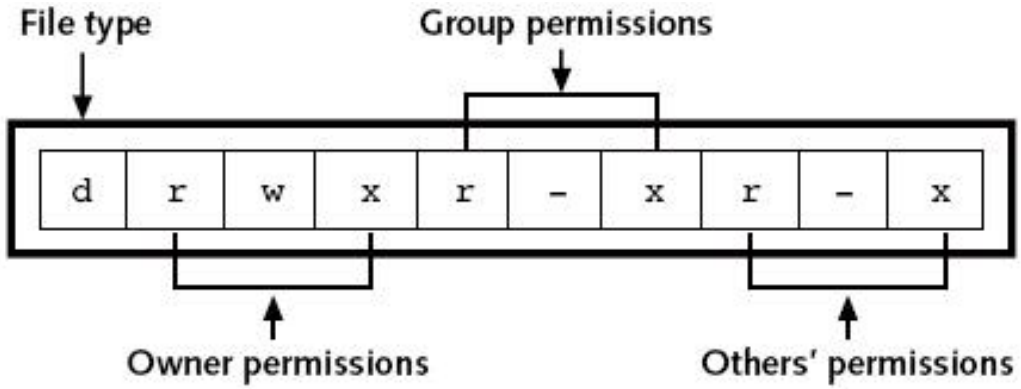

Figure 2-7 Example of the file type and the file permissions for a file

Syntax **chmod** [-option] *mode filename*

Dissection

- The argument can include the mode (permissions) and must include the file name. You can also use a wildcard to set the permissions on multiple files.
- Permissions are applied to owner (u), group (g), and others (o). The permissions are read (r), write (w), and execute (x). Use a plus sign (+) before the permissions to allow them or a hyphen (-) to disallow permissions. Octal permissions are assigned by a numeric value for each owner, group, and others.

Configuring File Permissions for Security (continued)

- The system administrator assigns group ids when he or she adds a new user account
  - A **group id (GID)** gives a group of users equal access to files that they all share
- Using *chmod* to change permissions of a file:

```
chmod ugo+rwx myfile
chmod go-wx account_info
```

  - Or, use the octal permission format

```
chmod 711 data
chmod 642 data
```

- 0 is no permissions.
- 1 is execute (same as *x*).
- 2 is write (same as *w*).
- 3 is write and execute (same as *wx*).
- 4 is read (same as *r*).
- 5 is read and execute (same as *rx*).
- 6 is read and write (same as *rw*).
- 7 is read, write, and execute (same as *rwx*).

Table 2-5 Suggestions for setting permissions

Type of File or Directory	Permissions Suggestion
System directories such as /bin, /boot, /dev, /etc, /sbin, /sys, and /usr	Give all permissions to root (the owner), rx to group and others—*chmod 755*.
/root directory for the root account	Give all permissions to root (the owner), rx to group, and no permissions to others—*chmod 750*.
Your home directory	Give all permissions to owner (your account), x or no permissions to group, and no permissions to others—*chmod 710* or *chmod 700*. (If you are a student and need to give your instructor access to your home directory, consider using *chmod 705* so your instructor has rx permissions.)
A subdirectory under your home directory that you want to share with others so they can access and create files	Give all permissions to owner (your account), group, and others—*chmod 777*.
A file in your home directory that you want people to be able to view, but not change	Give all permissions to owner (your account), rx to group, and rx to others—*chmod 755*.
A file that should only be accessed by you	Give all permissions to owner and no permissions to group and others—*chmod 700*.
An archived file in your home directory that should not be changed (just preserved) and that only you should be able to view	Give rx permissions to owner and no permissions to group and others—*chmod 500*.

Configuring File Permissions for Security (continued)

- **Sticky bit:** t (used in place of x)
  - Before: caused executable program to stay resident in memory after it was exited
  - Now: enables file to be executed, but only the file's owner or root have permission to delete or rename it
- **Set user id (SUID) bit:** s (used in place of x)
  - Gives current user temporary permissions to execute program-related files as though they are the owner
- **Set group ID (SGID) bit:** s (used in place of x)
  - Similar to SUID, but applies to groups

Summary

- In UNIX/Linux, a file is the basic component for data storage
- A file system is the UNIX/Linux systems' way of organizing files on storage devices
- The standard tree structure starts with the root (/) file system directory
- The section of the disk that holds a file system is called a partition
- A path, as defined in UNIX/Linux, serves as a map to access any file on the system

Summary (continued)

- You can customize your command prompt to display useful information
- The *ls* command displays the names of files and directories contained in a directory
- Wildcard characters can be used in a command and take the place of other characters in a file name
- Use *mkdir* to create a new directory
- Use *cp* to copy a source file to a destination file
- Use *chmod* to set permissions for files that you own

Command Summary

Command	Purpose	Options Covered in This Chapter
cd	Changes directories (with no options, *cd* goes to your home directory)	. Changes to the current working directory. .. Changes to the parent directory.
chmod	Sets file permissions for specified files	+ assigns permissions. -removes permissions.
cp	Copies files from one directory to another	**-b** makes a backup of the destination file, if an original one already exists (so you have a backup if overwriting a file). **-i** prevents overwriting of the destination file without warning. **-u** overwrites an existing file only if the source is newer than the file in the current destination.
ls	Displays a directory's contents, including its files and subdirectories	**-a** lists the hidden files. **-l** (lowercase L) generates a long listing of the directory. **-r** sorts the listing in reverse order. **-S** sorts the listing by file size. **-t** sorts by the time when the file or directory was last modified. **-X** sorts by extension.

Command Summary (continued)

Command	Purpose	Options Covered in This Chapter
mkdir	Makes a new directory	**-v** verifies that the directory is made.
mount	Connects the file system partitions to the directory tree when the system starts, and mounts additional devices, such as the CD/DVD drive	**-t** specifies the type of file system to mount.
rm	Removes a file	**-i** prompts before you delete the file.
rmdir	Removes an empty directory	**-v** provides a message to verify the directory is removed.
umask	Sets file permissions for multiple files	
umount	Disconnects the file system partitions from the directory tree