11.5.5 Linux File Management Facts

The file system determines how a computer's files are organized on a hard drive. Linux supports many different file system types. The following table describes several common file systems:

Type	Description
ext2	The Second Extended File System (ext2) is one of the oldest Linux file systems still available. - ext2 stores data in a standard directory and file hierarchy. - The maximum file size supported is 2 TB. - An ext2 volume can be up to 4 TB in size. - File names can be up to 255 characters long. - Linux users, groups, and permissions are supported. - It does not use journaling (which is used in most modern file systems). As a result, ext2 takes a long time to recover if the system shuts down abruptly.
ext3	The Third Extended File System (ext3) is an updated version of ext2 that supports journaling. Before committing a transaction to a storage device, the ext3 file system records the transaction to the journal and marks it as incomplete. After the disk transaction is complete, the file system marks the transaction as complete in the journal. By doing this, ext3 can keep track of the most recent file transactions and whether or not they were completed. This allows ext3 to recover much more quickly than ext2 in the event of an unclean system shutdown.
ReiserFS	The Reiser file system (ReiserFS) is an alternative to the ext3 file system. Like ext3, Reiser uses journaling to make crash recovery very fast. However, Reiser is a completely different file system from ext2 and ext3, using a dramatically different internal structure. ReiserFS supports a maximum file size of 8 TB and maximum volume size of 16 TB. In addition, the structure of Reiser allows it to perform much faster than ext2 or ext3.
ext4	ext4 is the fourth generation file system in the ext file system family. ext4 includes all of the features found with ext2 and ext3, with the addition of the following features: - Support for file sizes up to 16 TB and disk sizes up to 1 exabyte (EB) - Allows for up to four billion files in the file system - Uses checksums to verify the integrity of the journal file itself Checksums help improve the overall reliability of the system because the journal file is the most heavily used file of the disk.
swap	A swap file system is used as virtual memory (the portion of the hard disk used to temporarily store portions of main memory) by the operating system. A recommended practice is to make the swap file size between 1 and 1.5 times the amount of memory on the computer.
CDfs	CDfs is a virtual Linux file system that provides access to individual data and audio tracks on compact discs (CDs). A compact disc mounted with the "CDfs" driver appears as a collection of files, each representing a single track.
NFS	Network File System (NFS) is a distributed file system protocol that allows a user on a client computer to access files over a computer network much like local storage is accessed.
NTFS	Microsoft operating systems use NTFS (New Technology File System). Linux provides limited support for NTFS.
VFAT	VFAT is a FAT32 file system for Linux and does not support journaling. VFAT includes long name support. Support for VFAT must be compiled into the kernel for the system to recognize the VFAT format.
XFS	The XFS file system was developed for the Silicon Graphics IRIX operating system. An XFS file system is proficient at handling large files, offers smooth data transfers, and provides journaling. It also can reside on a regular disk partition or on a logical volume.
Btrfs	Btrfs is a Linux file system that uses a *copy-on-write* file system. Using copy-on-write technology, Btrfs provides several key features not found in earlier file systems: - **Storage pools** - Instead of using traditional disk partitions, Btrfs allows you to create *storage pools* from the storage devices in your system. From the storage pool, you can then allocate space to specific *storage volumes*. Instead of mounting partitions, you mount storage volumes at mount points in the file system. - **Snapshots** - The snapshot functionality provided by Btrfs protects data. It can be configured to take snapshots of your data at specified intervals and save it on separate media. If a file ever gets lost or corrupted, you can restore a previous version of the file from a snapshot.

Use the shell commands listed in the following table to manage the file system on Linux:

Command	Description
pwd	Displays the current directory on the screen.
ls	Displays a list of files and subdirectories that exist within a directory. Some options commonly used with the ls command include the following: - **-a** displays all files, including hidden files. - **-l** displays a detailed (long) listing of directory contents including ownership, permissions, modification dates, and file sizes. - **-R** displays the contents of the directory as well as all of its subdirectories.
cd	Changes directories in the file system. For example, to change to the /home directory in the file system, you would enter **cd /home** at the shell prompt.
cp	Copies files and directories from one location in the file system to another. For example, to copy the widget.odt file to the /home/rtracy directory, you would enter **cp widget.odt /home/rtracy** at the shell prompt. To copy an entire directory structure, include the **-R** option, which specifies that the directory contents be recursively copied.
mv	Moves files and directories from one location in the file system to another. For example, to move the widget.odt file to the /home/rtracy directory, you would enter **mv widget.odt /home/rtracy** at the shell prompt.
rm	Deletes files and directories from the file system. For example, to delete the widget.odt file, you would enter **rm widget.odt** at the shell prompt.
cat	Displays the contents of a text file on the screen. For example, to view the contents of the widget.txt file, you would enter **cat widget.txt** at the shell prompt.
less	Displays the contents of a text file on the screen, pausing the output one screen at a time. For example, to view the contents of the widget.txt file one page at a time, you would enter **less widget.txt** at the shell prompt.
head	Displays the first few lines of a text file on the screen. For example, to view the first lines of the widget.txt file, you would enter **head widget.txt** at the shell prompt.
tail	Displays the last few lines of a text file on the screen. For example, to view the last lines of the widget.txt file, you would enter **tail widget.txt** at the shell prompt. The **-f** option can be used with tail to monitor a file for changes. If new content is added to the end of the file (such as a log file), the new lines will be displayed on the screen.
vi	Edits the contents of a text file. The vi command uses four different operating modes: - Command mode - Command-line mode - Insert mode - Replace mode For example, to edit the contents of the widget.txt file, you would enter **vi widget.txt** at the shell prompt. You would then press the **Ins** key to enter Insert mode and make the necessary changes to the file. When done editing the file, you would press the **Esc** key to enter Command mode. The you would press the **:** key to enter command-line mode where you would enter **exit** to save your changes and exit the vi editor.
su	Allows you to switch user accounts. - **su -** (su with a space and a hyphen) is used to switch to the root user with the home directory and environment variables assigned to the root user.
chown	Changes the user or group that owns a file or directory. For example, to change the owner of the widget.txt file to the ksanders user, you would enter **chown ksanders widget.txt**.
chgrp	Changes the group that owns a file or directory. For example, to change the group that owns the widget.txt file to the RandD group, you would enter **chgrp RandD widget.txt**.
chmod	Changes the permissions assigned to three file system entities:

	- **Owner**: This is the user account that has been assigned to be the file or directory's owner. Permissions assigned to the owner apply only to that user account. - **Group**: This is the group that has been assigned ownership of the file or directory. Permissions assigned to the group apply to all user accounts that are members of that group. - **Others**: Identifies all other users who have successfully authenticated to the system. Permissions assigned to this entity apply to these user accounts. Files and folders in the Linux file system can be assigned one or more of the following permissions: - **Read**: Allows a file to be opened and viewed. Allows the contents of a directory to be listed. This permission is represented by a numeric value of **4**. - **Write**: Allows a file to be changed. Allows files to be added or deleted from a directory. This permission is represented by a numeric value of **2**. - **Execute**: Allows an executable file to be run. Allows a directory to be entered. This permission is represented by a numeric value of **1**. There are several different ways the chmod command can be used to modify the permissions assigned to a file or directory: - Enter **chmod *entity=permissions filename***. Substitute **u** for Owner, **g** for Group, and **o** for Others in the *entity* portion of the command. Substitute **r**, **w**, and/or **x** for the *permissions* portion of the command. - Enter **chmod *entity+/-permission filename***. Substitute **u** for Owner, **g** for Group, and **o** for Others in the *entity* portion of the command. Substitute **r**, **w**, and/or **x** for the *permissions* portion of the command. - Enter **chmod *3-digit_numeric_permission filename***. The first digit assigns permissions to Owner, the second to Group, and the third to Others.
dd	Copies file system data using records. It can copy files, directories, partitions, or even entire hard disks. For example to copy a disk partition to a file on a removable USB drive, you would enter **dd if=/dev/sdb1 of=/media/usb/partition.bak**.
shutdown	The **shutdown** command brings the system down or reboots the system in a secure manner. The syntax of the shutdown command is **shutdown +m -h|-r message**. The options for the shutdown command include: - **+m** specifies when to perform the shutdown operation. The amount of time is specified in minutes. - **-h** instructs the system to shut down and power down. - **-r** instructs the system to reboot after the shutdown. - **message** specifies a message that is sent to all users that accompanies the standard shutdown notification.

Use the **man *utility_name*** command at the shell prompt to view the syntax along with all of the options that can be used with these commands.