**2.1 Common Linux Shell Commands**

| Command | Function |
|---|---|
| **pwd** | Shows the present working directory. |
| **whoami** | Displays the current username. |
| **uname** | Prints system information. The **uname** command has the following options:<br>**-a** prints all system information.<br>**-o** prints the operating system.<br>**-p** prints the processor's architecture type. |
| **su** | Switches users in the shell prompt. The su command has the following options:<br>**su -l [username]** switches to the specified user and creates a new login shell.<br>**su [username]** (without the dash, but with the username) switches to the user in the current shell.<br>**su - [username]** (with the dash and the username) switches to the user and loads that user's environmental variables.<br>**su -** (with the dash but without the username) switches to the root user and loads the root user's environmental variables.<br>◦ The root user account is the Linux system superuser.<br>◦ The root user can perform any task. Some utilities do not work if the administrator is not logged in as the root user.<br>**su** (no dash or username) switches to the root user but does not load the root user's environmental variables.<br>**su** requires the password of the user except when switching from root to a normal user. |
| **exit** | Exits the current shell (which may close the login shell) or to go back to the original user after using the su command. |
| **exec** | Executes an executable to replace the shell process with the new process created by the executable file. |
| **cd** | Changes directories. For example, when the **/usr** directory is the current directory:<br>• **cd bin** changes to the **bin** directory in the current directory.<br>• **cd /usr/bin** changes to the **/usr/bin** directory from anywhere in the file system. |
| **ls** | Shows names of files and directories in the current directory. The **ls** command has the following options:<br>**-a** shows all files and directories, including hidden files.<br>**-l** shows extended information about files, including size, permissions, owner, and modified date.<br>**-d** displays only directories.<br>**-s** sorts files by size.<br>**-X** sorts by extension.<br>Many distributions use a color scheme to identify different file types as follows:<br>• Directories are blue.<br>• Text files are white.<br>• Links are cyan.<br>• Executable files are green.<br>• Compressed files are red. |
| **history** | Shows all the commands in the history queue. The **-c** option clears the history list.<br>History command queues are separate for each user. For example, a command typed as one user cannot be used after using the su command to switch to another user. |
| **clear** | Clears the shell screen. |
| **chsh** | Changes the default shell. The **chsh** command has the following options:<br>**-s** changes to a different installed shell. The command prompts for a password.<br>**-l** lists all installed shells.<br>For example, **chsh -s /bin/ksh [username]** changes the default shell for the user to the Korn shell if it is installed on the computer. |

**2.4 Alias**

| Command | Function |
|---|---|
| **alias** | Displays the currently defined aliases on the system. |
| **alias [name='command']** | Creates a custom alias that runs an existing command. A single alias can be defined to run multiple commands.<br><br>When creating the alias, encapsulate the command(s) with quotation marks or apostrophes. |
| **unalias [name]** | Removes an alias. |

**2.5 Environment Variables**

| Variable Type | Typical Source | Naming Convention | Inherited by Child Processes |
|---|---|---|---|
| **Environment** | Inherited from a parent process. It may be the system process that passes environment information gathered from a variety of system files and settings. | Uppercase | Yes |
| **Shell** | Generated by shell startup scripts. | Uppercase | No |
| **User** | Created by a user at the shell prompt or added when scripts that are defined in a user's profile are run. | Lowercase | No |
| **Local** | Defined in scripts and in script functions. | Lowercase | No |

**2.5 Persistent Environment Variables**

| File(s) | Description |
|---|---|
| **/etc/environment** | The /etc/environment file is a system-wide configuration file for all users. Environment variables defined in this file are available to all shells and processes. Changes to this file do not take affect until after a reboot. |
| **/etc/profile** | This file is run for all users, but only during interactive logins (when a username and password is required to log in). While environment variable can be defined in this file, the preferred method is to define them in a separate script file saved in the /etc/profile.d directory.<br>An interactive login occurs when the user must supply a username and password. An example is when the user connects using SSH, when using Ctrl+Alt+F2 to log in to a virtual terminal, or when using the su (switch user) command. |
| **/etc/profile.d/*.sh** | These script files are run for all users, but only during interactive logins. You can add environment variable definitions to these files. These files are run by the /etc/profile file. |
| **~/.bash_profile**<br>**~/.bash_login**<br>**~/.profile** | These hidden files are located in a user's home directory and are only run for that specific user and only during interactive logins. A search is made for these files in the order that is listed. Only the first file found will be run. Environment variable definitions can be added to these files. |
| **/etc/bashrc** | This file is run for all users during a Bash shell initialization. It is run during both interactive logins and interactive non-logins. Environment variable definitions can be added to them.<br>An interactive non-login occurs when the user is not required to enter a username and password to open a shell. An example is when the user opens a graphical terminal window in gnome or a new shell using the bash command. |
| **~/.bashrc** | This hidden file is located in a users home directory and is only run for that specific user during both interactive logins and non-interactive logins. You can add environment variable definitions to this file. |

**2.5 Commands Working With Environment Variables**

| Command | Function |
|---|---|
| **[name]=[value]** | Creates a new shell variable with an assigned value or changes the value of an existing variable. To append information to a variable instead of replacing it, include the current variable in the command. Changing the value of an environment variable will change it for the current shell session and any subsequent child processes. |
| **export [name]=[value]** | Creates a new environment variable for the current shell session and any subsequent child processes. |
| **export [name]** | Exports an existing shell variable to make it an environment variable for the current shell session and any subsequent child processes. |
| **declare -x [name]=[value]** | Creates a new environment variable for the current shell session and any subsequent child processes (functionally equivalent to export [name]=[value]). |
| **declare -x [name]** | Exports an existing shell variable to make it an environment variable for the current shell session and any subsequent child processes (functionally equivalent to export [name]). |
| **export -n [name]** | Removes the export property of an environment variable, making it a shell variable. |
| **declare +x [name]** | Removes the export property of an environment variable, making it a shell variable (functionally equivalent to export -n [name]). |
| **echo $[name]** | Displays the contents of an environment variable (or any variable). |
| **printenv** | Displays a list of the current environment variables. |
| **env** | Displays a list of the current environment variables.<br>You can also use the env command to run a command using temporarily manipulated environment variables. |
| **export -p** | Displays a list of all exported variables and functions. |
| **set** | Displays a list of all environment variables, shell variables, local variables, and shell functions. You can also use the set command to set or unset values of shell options and positional parameters. |
| **unset [name]** | Removes the variable and its value independent of whether the variable is an environment variable or a shell variable. |

**2.5 Common Environment Variables**

| Variable | Description |
|---|---|
| **SHELL** | The user's login shell. |
| **DISPLAY** | The location where Windows output is displayed. |
| **ENV** | The location of the configuration file for the current shell. |
| **HISTSIZE** | The number of lines or commands that are stored in a history list while your shell session is ongoing. |
| **HOME** | The absolute path of the user's home directory. |
| **HOSTNAME** | HOSTNAME is identical to HOST, but is only used on certain distributions. |
| **LOGNAME** | The current user's username. |
| **MAIL** | The path to the current user's mailbox file. |
| **OLDPWD** | The directory the user was in prior to switching to the current directory. |
| **PATH** | The directory paths used to search for programs and files.<br>Use a colon to separate entries in the PATH variable.<br>Do not include a period (.) in the PATH variable. A period indicates that the working directory is in the path and poses a security risk. |
| **PWD** | The path of the current working directory. |
| **LANG** | The language the operating system uses. |
| **TERM** | The type of terminal to emulate when running the shell. |
| **USER** | The current logged in user. |

**2.5 Bash Shell Variables**

| Variable | Description |
|---|---|
| **BASH** | The location of the Bash executable file. |
| **BASHOPTS** | The list of options that were used when Bash was executed. |
| **BASH_VERSION** | The Bash version being executed in a readable form. |
| **BASH_VERSINFO** | The Bash version in machine-readable output. |
| **EUID** | The current user's ID number. |
| **HISTFILE** | The filename where past commands are stored. |
| **HISTFILESIZE** | The number of past commands that HISTFILE stores for multiple sessions. |
| **OSTYPE** | The type of operating system (usually Linux). |
| **PS1** | The characters that the shell uses to define what the shell prompt looks like. |
| **COLUMNS** | The number of columns being used to draw output on the screen. |
| **LINES** | The number of lines being used to draw output on the screen. |
| **UID** | The current user's user ID. |

**2.6 Shell Configuration Files**

| Configuration File | Run by | Shell Type |
|---|---|---|
| **/etc/bashrc** or **/etc/bash.bashrc** | All users | Non-login |
| **~/.bashrc** | The specified user only | Non-login<br>On most Linux distributions, this file is also called by login shell configuration files. |
| **/etc/profile** | All users | Login |
| ***. sh files in the /etc/profile.d/ directory** | All users | Login<br>On most Linux distributions, this file is also called by non-login shell configuration files. |
| **~/.bash_profile** | The specified user only | Login |
| **~/.bash_login** | The specified user only | Login |
| **~/.profile** | The specified user only | Login |
| **~/.bash_logout** | The specified user only | Login<br>This file is only run as the user logs out |

**2.7 Redirecting input / output**

| Command Operator | Description |
|---|---|
| **>**<br>**1>** | Redirects command output that is normally sent to stdout to the file name that follows the operator. The 1 is implied so that > and 1> are functionally identical.<br>• If the file that follows the operator exists, it is overwritten.<br>• If the file doesn't exist, it is created.<br>• If there is no output generated by the command, the file will be empty. |
| **2>** | Redirects command output that is normally sent to stderr to the file name that follows the operator. The 2 is mandatory.<br>• If the file that follows the operator exists, it is overwritten.<br>• If the file doesn't exist, it is created.<br>• If there is no error generated by the command, the file will be empty. |
| **>>**<br>**1>>**<br>**2>>** | The >> operator functions in the same way as the > operator, except that any output/errors are appended to the file that follows the operator.<br>• The >> and 1>> operators append the output sent to stdout.<br>• The 2>> operator appends the output sent to stderr.<br>• If the file that follows the operator exists, it is appended with the output/error.<br>• If the file doesn't exist, it is created.<br>• The file will not be appended if there is no output/errors generated by the command. |
| **&>** | Redirects both command output that is normally sent to stdout and command errors that are normally sent to stderr to the file name that follows the operator.<br>• As part of a command, &> myfile.txt is equivalent to > myfile.txt 2> &1 or > myfile.txt 2> myfile.txt.<br>• File creation follows the rules for the > operator. |
| **&>>** | The &>> operator functions in the same way as the &> operator, except that both output and errors are appended to the file that follows the operator.<br>• As part of a command, &>> myfile.txt is equivalent to >> myfile.txt 2>> &1 (where & indicates that what follows is a file descriptor and not a filename) or >> myfile.txt 2>> myfile.txt.<br>• File creation follows the rules for the > operator. |
| **<**<br>**0<** | Redirects command input that is normally read from stdin so that it is read from the file name that follows the operator. The 0 is implied so that < and 0< are functionally identical.<br>• If the file that follows the operator exists, it is used as input.<br>• If the file doesn't exist, an error is shown.<br>• If there is no input needed by the command, the file is ignored. |

| *Standard Stream* | *File Descriptor* | *Associated Device* |
|---|---|---|
| **stdin** | 0 | Keyboard |
| **stdout** | 1 | Console screen or graphical shell window |
| **stderr** | 2 | Console screen or graphical shell window |

**2.7 Piping**

**Piping Information & Facts**

Piping redirects the output from one command to be the input of another command.

- A Linux pipe is represented by a vertical bar (**|**).
- The pipe functionality is similar to that of using stdout redirection to write the output of one command to an intermediate file, that is then used as input to a second command, using stdin redirection.
- A plumbing pipe, where water enters from one end and exits the other, can be used as a mnemonic to help explain how the pipe (**|**) operator works.

**2.7 The tee Command**

**"tee" Information & Facts**

There may be times when you want to view the output of a command as it is normally sent to the console screen (stdout), but you also want the same output to be saved in a file. This can be done using the tee command

- The output from a command is piped to the tee command.
- The file used to store the output is added as a tee command argument.
- A plumbing tee, where water flow is divided from one pipe to two separate pipes, can help you understand how the tee command works.

*Example* *Result*

**ls /bin | tee binfiles.txt**

Displays the files and directories contained in the /bin directory on the console screen (or shell window) and writes the same information to the binfiles.txt file.

**ls -1 *.txt | wc -l | tee count.txt**

Pipes a one-column list of files that end with .txt in the current directory to the wc (word count) command and then take that output, which gives the number of files in the list, and pipes that to the tee command that displays this number on the console screen (or shell window) and writes the same number to the count.txt file.

**2.7 Commonly used with Redirection & Piping**

**Device Files**

| | |
|---|---|
| **/dev/tty** | The first terminals were Teletype (abbreviated as tty), which can be compared to a remote-controlled typewriter. The /dev/tty device file is associated with the computer's controlling terminal or the shell's window.<br>• Data can be both written to and read from this file.<br>• Text written to this file is displayed on the console monitor's screen or shell window.<br>• Text read from this file originates from the console's keyboard.<br>• The /dev/tty device file is similar to a combination of stdin and stdout. Both stdin and stdout are accessed as data streams, whereas /dev/tty is accessed like a file. |
| **/dev/null** | The /dev/null device file is associated with a null device. A null device is commonly used for disposing of unwanted output streams.<br>• While a command can read from /dev/null, commands typically write unwanted output or unwanted error messages to /dev/null.<br>• A slang word for the /dev/nul device file is bit bucket. |
| **/dev/zero** | Similar to /dev/null, /dev/zero discards any input. It also returns a "0" for however many times it is accessed. It is most commonly used for:<br>• Initializing a new block device<br>• Overwriting existing data |
| **/dev/urandom** | Returns a pseudo-random number. Frequently used when performing cryptographic (encryption) tasks. |

**2.7 Command Substitution**

**Command Substitution Information**

Command substitution is a feature of the Bash shell that substitutes the output of one shell command as the arguments for another shell command.

Command substitution is implemented with the **$(<command>)** operator. The following occurs when the shell's command line interpreter encounters the **$()** operator:

1. The shell creates a child process that runs the specified command.
2. The stdout from the first command is redirected back to the shell.
3. The shell parses the output from the first command into words separated by white space.

If the command substitution operator $() appears within double quotes ("), word parsing is not performed on the output.

4. The shell creates a new command by substituting the parsed output from the first command in place of the **$(<command>)** operator.
5. The shell creates another child process that runs the second command.

The backtick (`) operator also performs command substitution. Enclose a command within backticks (`) to achieve command substitution in the same way as the **$()** operator. On newer keyboards, the backtick (`) is on the same key as the tilde (**~**).

| Example | Explanation |
|---|---|
| **printf "The date and time is: $(date)\n"** | Command substitution occurs when the shell encounters **$(date)**. The process is as follows:<br>**1.** The shell creates a child process and runs the **$(date)** command.<br>**2.** The output from the child process is redirected back to the shell but is not parsed.<br>**3.** The **$(date)** operator in the original command is replaced with the output from the child process.<br>**4.** Another new process is created that runs the **printf** command with the replaced text.<br>The **printf** command replaces the **\n** in double quotes with a newline character. |
| **echo -e "List of logged on users and what they are doing:\n $(w)"** | Command substitution occurs when the shell encounters **$(w)**. The process is as follows:<br>**1.** The shell creates a child process and runs the **w** command.<br>The **w** command is short for "who" and returns a summary of logged-on users.<br>**2.** The output from the child process is redirected back to the shell but is not parsed.<br>**3.** The **$(w)** operator in the original command is replaced with the output from the child process.<br>4. Another new process is created that runs the **echo** command with the replaced text.<br>The **-e** option with the **echo** command causes the **\n** in double quotes to be replaced with a newline character. |

**2.7 Command Substitution**

**The xargs Command**

Linux commands have a maximum length of 128 KB. Command substitution will give an error if the command that is formed after the substitution is over 128 KB. One solution is to pipe the first command to the **xargs** command. The **xargs** command breaks down the output from the first command into 128-KB chunks. Each chunk is passed in one at a time as an argument to the **xargs** command.

| Example | Explanation |
|---|---|
| **find /home -name *~ \| xargs rm** | This command deletes all files in all subdirectories of the /home directory that end with the tilde (~) character.<br>**1.** The **find /home -name *~** command returns a list of all the files in all the subdirectories under the /home directory that end with the tilde (~) character.<br>**2.** The file list is piped as a stream to the xargs command.<br>**3.** The xargs command collects the first 128-KB chunk from the stream.<br>**4.** The rm command is run using the 128-KB chunk as an argument.<br>**5.** The xargs command continues to collect 128-KB chunks from the stream and continues to run the rm command using the chunks until the end of the stream. |
| **ls -S *.txt \| xargs wc** | When there are large number of *.txt files in a directory, this command displays a list of all the files along with the number of lines/words/characters in each. The list is sorted by size, with the largest files shown first.<br>**1.** The **ls -S *.txt** command returns the list of files sorted from largest to smallest.<br>**2.** The file list is piped as a stream to the **xargs** command.<br>**3.** The **xargs** command collects the first 128-KB chunk from the stream.<br>**4.** The **wc** command is run using the 128-KB chunk as an argument.<br>**5.** The **xargs** command continues to collect 128-KB chunks from the stream and continues to run the **wc** command using the chunks until the end of the stream. |

**2.9 File Viewing**

| Command | Description |
|---|---|
| **cat** | Displays the contents of a file in the shell. This command can display multiple files at once. |
| **less** | Displays the contents of a file, pausing one screen at a time.<br>• Use the Spacebar to scroll to the next screen.<br>• Use the Up arrow and Down arrow to scroll up and down.<br>• Press q to exit. |
| **head** | Lists the first ten lines (the default) of a specified file. Use the -n option to specify a specific number of lines to display. |
| **tail** | Lists the last ten lines (the default) of a specified file.<br>**-n** specifies a specific number of lines.<br>**-f** monitors the file. |

**2.9 File Management**

| Command | Function |
|---|---|
| **touch** | If the file does not exist, touch creates a blank version of the file. If the file does exist, this command updates the file's modification and last accessed time. |
| **file** | Shows the file type. The file command is useful because Linux does not require file extensions. The file command uses file signatures in:<br>• /usr/share/misc/magic<br>• /usr/share/misc/magic.mgc<br>• /etc/magic |
| **stat** | Displays file or file system status. Useful for tracking changes to files or creation dates.<br>**-f** shows file system status instead of file information.<br>**-L** follows links to the target file before displaying information. |
| **cp** | Copies files. Copying leaves the source file intact.<br>**-f** overwrites files that already exist in the destination directory.<br>**-i** prompts before overwriting a file in the destination directory. |
| **mv** | Moves or renames files (and directories). Moving files erases the source file and moves it to the destination.<br>**-f** overwrites files that already exist in the destination directory.<br>**-i** prompts before overwriting a file in the destination directory.<br>**-n** never overwrites files in the destination directory. |
| **rm** | Removes a file or directory.<br>• Use **-i** to be prompted before deleting the file.<br>• Use **--dir** to remove empty directories.<br>• Use **-r** to remove directories and their contents recursively.<br>• Use the **-f** option to not prompt you when there are nonexistent files or arguments used with the command.<br>The rm command deletes a file or directory's inode, but it does not actually delete its data. To permanently remove data, use the shred command. |
| **shred** | Deletes the file and overwrites the file's data on the hard disk. The shred command is useful when deleting files that contain proprietary information or other sensitive data.<br>**-n** specifies the times to overwrite. The default is 25 times.<br>**-u** deletes the inode.<br>**-v** displays the progress of the file deletion.<br>**-z** overwrites the filename with zeros. |
| **chattr** | Modifies file attributes. A + or - is used to add or remove attributes, respectively. For example, to make a file immutable (cannot be modified), using the +i flag sets the immutable file attribute, and -i removes the immutable file attribute.<br>**-R** recursively changes attributes of directories and their content.<br>**-V** displays verbose output and the program version.<br>**-f** suppresses error messages.<br>**-p** sets the file's project number.<br>**-v** sets the file's version/generation number. |
| **lsattr** | Lists file attributes.<br>**-R** recursively lists attributes of directories and their content.<br>**-V** displays the program version.<br>**-a** lists all files in directories.<br>**-d** lists directories like other files rather than listing their content.<br>**-v** lists the file's version/generation number. |

**2.10 Link Facts**

| Type | Description |
|---|---|
| **Hard link** | A hard link is a duplicate entry in the file system that points to a specific piece of data on the disk drive. With a hard link:<br>• The same inode is used as the original file. The inode specifies where a file's data physically exists on a disk.<br>The **ls -i** command displays the inodes for the files and directories in a directory.<br>• No data is stored in a hard link. The data is pointed to by the inode, which the hard link refers to.<br>• Files are basically hard links when created, pointing to the inode where the data is stored. Until all hard links that point to an inode and its data are deleted, the data will not be removed.<br>• In the output from th**e ls -a** command, a hyphen is used as the first character in the permission string for a hard link, which is the same character used for normal files (for example, -rwxr-xr-x). |
| **Symbolic link "soft link"** | A symbolic link (also known as a soft link) is a file that points to another file in the file system. A symbolic link is similar to shortcuts in the Windows OS. With a symbolic link:<br>• Separate inodes are used. The link file has an inode that is distinct from the inode of the file being pointed to.<br>• In the output from the ls -a command:<br>◦ A lower-case L (l) is used as the first character in the permission string (for example, lrwxrwxrwx indicates a symbolic link).<br>◦ The -> character sequence follows the file name, which is followed by the file that the link points to. |

**2.10 Creating Links**

| Command | Function |
|---|---|
| **ln [source] [link_name]** | Creates links.<br>• **ln** (with no options) creates a hard link between files.<br>• **ln -s** creates a symbolic link to a file.<br>**Examples**<br>**ln /home/jsmith/project1 /home/edunford/project1**<br>*creates a hard link to /home/jsmith/project1 in /home/edunford/*<br>**ln -s /home/jsmith/project1 /home/edunford/project1_ln**<br>*creates a symbolic link named /home/edunford/project1_ln that points to /home/jsmith/project1* |
| **cp [source] [link_name]** | Copies files and creates links.<br>• **cp -l** creates hard links rather than copying the files.<br>• **cp -s** creates symbolic links rather than copying the files.<br>**Examples**<br>**cp -l /home/jed/fil1 /home/esam/proj1**<br>*creates an exact copy of /home/jed/fil1 in /home/esam/*<br>**cp -s /home/mkon/text /home/ytew/text_ln**<br>*creates a symbolic link named /home/ytew/text_ln that points to /home/mkon/text* |
| **unlink [link_name]** | Removes both symbolic links and hard links. |

**2.11 FHS Directories**

| Directory | Description |
|---|---|
| **/** | The / character represents the root directory of the Linux system. All other directories are located beneath the / (root directory) of the system. |
| **/bin** | The /bin directory contains binary commands that are available to all users. |
| **/boot** | The /boot directory contains the kernel and bootloader files. |
| **/dev** | The /dev directory contains device files that represent the hardware used by the system, such as a hard drive, mouse, and printer. |
| **/etc** | The /etc directory contains configuration files specific to the system. |
| **/home** | The /home directory contains (by default) the user home directories. |
| **/lib** | The /lib directory contains shared program libraries and kernel modules. |
| **/media** | The /media directory is used to mount removable media, such as optical discs and USB drives. |
| **/mnt** | The /mnt directory is used for temporarily mounting local and remote file systems. |
| **/opt** | The /opt directory contains additional programs on the system. |
| **/proc** | The /proc directory is a virtual directory (only exists in RAM) that contains information about the system state and processes. |
| **/root** | The /root directory is the root user's home directory. Do not confuse /root with the root of the file system (/). |
| **/sbin** | The /sbin directory contains system binary commands. |
| **/srv** | The /srv directory contains files for services such as HTTP and FTP servers. |
| **/sys** | The /sys directory contains the sysfs virtual file system which displays information about devices and drivers. |
| **/tmp** | The /tmp directory contains temporary files created by programs during system use. |
| **/usr** | The /usr directory contains system commands and utilities. |
| **/var** | The /var directory contains data files that change constantly, such as:<br>• User mailboxes<br>• Print queues<br>• Log files |

**2.12 File Searching**

| Command | Description |
|---|---|
| **find** | Searches through all files based on the file system by name, file size, time created, and other options. Be aware of the following find command options:<br>**-name** locates a file or directory by name in a specific path. When using -name:<br>• Enclose name strings in single quotes.<br>• Use wildcards for partial names.<br>**-iname** same as **-name** but ignores case.<br>**-user** finds files owned by a specific user.<br>**-size** finds files of a specific size. Use the following options:<br>**c** for bytes<br>**k** for kilobytes<br>**M** for megabytes<br>**-mtime** finds files last modified before or after a specified number of days ago.<br>**-type** specifies whether to find files or directories.<br>**f** for files<br>**d** for directories<br>**-maxdepth** specifies how many levels down to search.<br>**-print0** finds filenames with spaces.<br>**-o** specifies the or parameter when searching with multiple criteria.<br>**. (period)** specifies the search locations as the current directory and subdirectories. |
| **locate** | Searches for files in the file system. The locate utility maintains a database containing a listing of all the files and directories in the file system. Be aware that the locate command:<br>• Searches through an index instead of the actual file system. Because of this, the locate command usually runs much faster than the find command.<br>• Searches all files if no path is specified.<br>• Maintains an index of all files and directories in the */var/log/locatedb* file.<br>• Uses the updatedb command to update the index. (The */etc/updatedb.conf* file is used to configure for updatedb.)<br>• The index is updated on a regular schedule. This means it is possible for locate to find a file in the index that no longer exists in the actual file system. Likewise, it is possible for locate not to find a new file in the file system that has not been added to the index yet.<br>Be aware of the following locate command options:<br>**-c** counts the number of entries rather than list them.<br>**-e** lists files only after verifying that they exist.<br>**-i** ignores case.<br>**-l** limits the number of files listed.<br>**-b** searches for the string in only file or directory base names. |
| **which** | Displays the path to a command and determines whether a package is installed. |
| **whereis** | Displays the path to a Linux command's binary files, manual pages, and source code (if sources are installed). Be aware of the following whereis command options:<br>**-b** lists the path to the binary file (similar to which).<br>**-m** lists the path to the man page files.<br>**-s** lists the location of the source code.<br>**-u** lists entries that do not have source code, binary file, and man page locations.<br>When no options are specified, whereis shows all available data. |
| **type** | Displays a command's type. Possible categories include:<br>• A built-in shell command<br>• A command that the shell calls<br>• An aliased command<br>• A function<br>If a called command has been used recently, the output says that the command is hashed, which means that it is in the shell's hash table. |

**2.12 File & Directory Searching Wildcards**

| Metacharacter | Description |
|---|---|
| * | Matches any number of any characters |
| . | Matches a single character |
| ^ | Matches an expression if it appears at the beginning of a line |
| $ | Matches an expression if it appears at the end of a line |

**2.12 File Content Searching & Comparison**

| Command | Description |
|---|---|
| **grep** | Searches through files for a specified character string. By default, grep is context sensitive and displays the string in the context of the line containing the string.<br>**-A [number]** prints a specified number of lines following the matching lines.<br>**-a** searches binary (executable) files as though they were text files.<br>**-B [number]** prints a specified number of lines before the matching lines.<br>**-C [number]** prints a specified number of lines of context around the matching lines.<br>**-c** shows the number of matches of the string for the file.<br>**-E** uses regular expressions for the text pattern.<br>**-e [pattern]** specifies a literal pattern.<br>**-f** searches for multiple strings using a file that lists the string patterns.<br>**-i** ignores case sensitivity in a search string.<br>-l lists just the names of the files with a match. This is used to search multiple files.<br>**-m [number]** shows only a specified number of matches for a file.<br>**-n** displays the line number of the lines containing the term.<br>**-r** searches the directory and all subdirectories for files containing the term.<br>**-v** displays non-matching lines.<br>**--include=[file_name]** searches only in files with names that match a specified string.<br>**--exclude=[file_name]** searches in files with names that do not match a specified string.<br>**-w** searches for whole words only. |
| **egrep** | Uses regular expressions in the search strings. The egrep command uses the same options and syntax as grep and is identical to **grep -E**. Constructors for egrep regular expressions include:<br>**^** matches terms that occur at the beginning of a line.<br>**$** matches terms that occur at the end of a line.<br>**\<** matches words that begin with the term.<br>**\>** matches words that end with the term.<br>**[asdf]** matches any one of the characters in the brackets.<br>**[0-9]** matches any of the range of numbers 0-9.<br>**[^xyz]** omits any one of the letters in the list<br>**.** matches any single character.<br>**[asdf]+** matches one or more of the characters in the list.<br>***** matches any number or none of the preceding single character.<br>**\|** matches either of the terms.<br>**** displays the literal value of a character used for expressions.<br>**()** groups expressions. |
| **fgrep** | Uses a file as the source for the string patterns. When searching for fixed strings rather than regular expressions, fgrep:<br>• Uses the same options as the grep command and has the same syntax.<br>• Is identical to grep -F, but searches faster than grep.<br>• Interprets the pattern as a list of fixed strings, any of which can be matched.<br>**fgrep [options] [ -e pattern_list] [pattern] [file]**<br>**-c** It is used to print only a count of the lines which contain the pattern.<br>**-h** Used to display the matched lines.<br>**-i** During comparisons, it will ignore upper/lower case distinction.<br>**-l** Used to print the names of files with matching lines once, separated by new-lines. It will not repeat the names of files when the pattern is found more than once.<br>**-n** It is used precede each line by its line number in the file (first line is 1).<br>**-s** It will only display the error messages.<br>**-v** Print all lines except those contain the pattern.<br>**-x** Print only lines matched entirely.<br>**-e pattern_list** Search for a string in pattern-list (useful when the string begins with a "-").<br>**-f pattern-file** Take the list of patterns from pattern-file.<br>**pattern** Specify a pattern to be used during the search for input.<br>**file** A path name of a file to be searched for the patterns. If no file operands are specified, the standard input will be used. |

| | |
|---|---|
| **diff** | Displays the differences between two files, line by line. The output can contain the following characters:<br>**< [text]** only the first file contains this text.<br>**> [text]** only the second file contains this text.<br>**a** text has been added.<br>**c** text has changed.<br>**d** text has been deleted.<br>Options for the diff command include the following:<br>**-c** displays differences in context mode.<br>**-u [number]** prints a specified number of lines in a unified context.<br>-I ignores the case and treats uppercase and lowercase the same.<br>**-w** ignores all white space.<br>**-y** displays the output in two columns. |

**2.13 Text Stream Processing**

| Command | Function |
|---|---|
| **cut** | Prints just the columns or fields that you specify to the standard output. By default, the tab character is used as a delimiter to define each field. Options include the following:<br>**-c** cuts characters.<br>**-f** cuts fields.<br>**-d** specifies the character used as the field delimiter. The default is a tab.<br>**-s** removes lines that do not have a field delimiter.<br>**-d' '** specifies a space as the field delimiter. |
| **expand** | Replaces a tab character with a specified number of spaces. The default is eight spaces.<br>**-t** specifies the number of spaces to be used. |
| **fmt** | Formats lines in a file or text stream to a uniform length. This is useful to format long lines of text to fit in a terminal window. Options include:<br>**-w** specifies the number of characters for the width. The default is 75.<br>**-s** prevents the command from formatting lines shorter than the specified length. This command is often used with code text to keep lines of code separate. |
| **join** | Combines text from two files based on identical fields and sends the result to standard output. By default, fields are offset by whitespace. Options include the following:<br>**-i** ignores case when searching for identical text.<br>**-j** specifies the number of the field to use when joining. This specifies both files.<br>**-1** specifies the number of the field from the first listed file to use when joining.<br>**-2** specifies the number of the field from the second listed file to use when joining.<br>**-t** specifies the character to use as the field delimiter. |
| **nl** | Places a line number in front of each line in a text file and sends the result to standard output. Options include the following:<br>**-i** specifies the increment to use when numbering the lines.<br>**-v** specifies the starting number.<br>**-s** specifies the text to be placed between the number and the line. the default is two spaces. |
| **od** | Displays the contents of any file in octal, decimal, hexadecimal, or character format. Options include the following:<br>**-b** specifies an octal dump.<br>**-d** specifies a decimal dump.<br>**-x** specifies a hexadecimal dump.<br>**-c** specifies a character dump. |
| **paste** | Adds the contents of one file to the contents of another file on a line-by-line basis.<br>• By default, the tab character is used to separate columns.<br>**-d** specifies a character to place between the conjoined lines of each file. Only a single character can be specified. |
| **pr** | Formats a text file for printing. By default, this command:<br>• Separates files into 66-line pages.<br>• Uses the first five lines to create a header that contains a page number, the time and date, and the path to the file.<br>• Uses the last five lines to create a footer of blank lines.<br>Options include the following:<br>**-d** double-spaces the lines.<br>**-h** specifies text to replace the file name in the header.<br>**-l** specifies the number of lines. The default is 66.<br>**-t** prevents the command from creating the header and footer.<br>**-o** creates a margin on the left side of the text. |

| | |
|---|---|
| **sed** | Takes text or commands from the command line as input and modifies the text document named in the command line. sed is particularly useful under the following circumstances:<br>• When a file is too large to open and edit conveniently in a text editor.<br>• When the series of edits (for example, adding line spacing, margins, replacing text) is too complex to perform easily in a text editor.<br>• When it is easier to perform a series of global document changes.<br>Flags and options include the following:<br>**s** replaces the text behind the first / with the text behind the second /. To save the results of the command, use > to redirect the output to a new file.<br>**d** deletes lines that contain the specified term.<br>**g** changes all occurrences of the term in a line.<br>**p** prints the modified lines in addition to the standard output.<br>**-n** suppresses all printing. The p flag can be used to print the modified lines.<br>**-e** allows multiple commands in as sed operation.<br>**-f** calls a file filled with editing commands (one command per line) to perform a number of operations at one time instead of doing them individually from the command line. |
| **awk** | Creates reports based on the data you retrieve from files, builds databases, or performs mathematical operations against numbers in text files.<br>Be aware of the following patterns and actions:<br>**-f** specifies a file containing awk commands to be used.<br>**-F** specifies the field delimiter to be used. The default is whitespace.<br>**$#** is used to designate fields. For example, $6 is the sixth field in a line.<br>**\t** inserts a tab.<br>**\n** inserts a newline character.<br>**\f** inserts a form-feed character.<br>**\r** inserts a carriage return. |
| **sort** | Sorts each line of text in a file or from a text stream alphabetically. Options include the following:<br>**-b** ignores leading blank spaces.<br>**-d** uses the first alpha-numeric character and ignores special characters.<br>**-f** ignores case.<br>**-M** sorts by month.<br>**-n** sorts according to the string numeric value.<br>**-r** reverses the sort order. |
| **split** | Splits lines of text from a file or a text stream into segments of a specified number of lines.<br>Options include:<br>**-l** specifies the number of lines per file.<br>**-number** " "<br>**-b** splits text into a specified byte size instead of a number of lines.<br>**-d** uses numeric suffixes rather than alphabetic.<br>**-a** specifies the number of characters in the suffix. |
| **tr** | Transposes characters in a text stream. tr only works with character streams. The command uses two character sets.<br>• The first set specifies the characters to be changed.<br>• The second set specifies what they should be changed to.<br>Options include the following:<br>**-c** changes all characters except those specified in the first set.<br>**-d** deletes characters found in the first set.<br>**-s** changes double characters to single ones.<br>**-t** truncates the first set of characters to match the size of the second set. |
| **printf** | Creates formatted output by inserting arguments into user-defined strings of text. |
| **unexpand** | Changes spaces into a tab. Options include the following:<br>**-a** specifies that the command change all occurrences. Without -a, the command only changes leading spaces.<br>**-t** specifies the number of spaces to be changed. The default is eight. |
| **uniq** | Filters identical lines from a file. The lines must be adjacent. Options include the following:<br>**-d** prints only the duplicate lines.<br>**-f** specifies the number of initial words to skip. Words are delimited by white space.<br>**-s** specifies the number of initial characters to skip.<br>**-w** specifies the number of characters to compare in each line.<br>**-u** leaves out the duplicate lines. |
| **wc** | Prints the number of bytes, characters, lines, or words, or the length of the longest line from the text of a file or text stream. Options include the following:<br>**-c** specifies bytes.<br>**-m** specifies characters. Character count is often identical to byte count.<br>**-l** specifies line count.<br>**-L** specifies the length of the longest line.<br>**-w** specifies word count.<br>When no options are used, the command prints line count, word count, and byte count, respectively. |