Using the Shell

Linux commands are case sensitive. Running the `ls` command is not the same as `LS` or `Ls`.
The `uname` command displays information about the current system.

1. Short options are specified with a hyphen - followed by a single character.
2. Long options for commands are preceded by a double hyphen --and the option is typically a "full name".

[You will notice that with the `-a` option specified, the result now includes hidden files that begin with a dot `.` character:]

The same effect can be achieved by using a long option `--all` with the `ls` command. Execute the following:
If you don't include two hyphens and instead use the option `-all`, your output will be different because you will really be executing the `ls -a -l` command.
Execute the `ls` command with the `-l` option; this will show the listing in long format:
The `-r` option for the `ls` command produces the listing in reverse sorting order. Compare it with the output from the `ls` command executed earlier.
`uname` command provides various system information, including system name (`-n`) and processor type (`-p`).
The `-w` option, when specified with the `ls` command, can be used to specify the width of the screen (normally this is determined by the actual terminal window size):`ls -rw 40` the `-w` option requires an argument, it must be specified at the end of the options.

`ls -w 40 -I "T*"` The ignore `-I` option will cause the `ls` command to not display specific files that match a pattern. The pattern `"T*"` means any file or directory that begins with a capital T. In this case, the `Templates` directory was not displayed.

Aliases serve as a nickname for another command or series of commands.`alias name=command`

you can remove entries from your list of aliases by using the `unalias` command:

The `type` command is used to determine what a file is and/or where it is located. Use the `type` command to get information about the `pwd` command.

f you do not know the exact man page you are looking for, you can search for man pages that match a keyword by using the `-k` option to the `man`command. For example, execute the following:**Note**: When you use the `-k` option, both the man page names and descriptions are searched.

To search only for the man page name, use the `-f` option: Alternatively, you can try the `whatis` command, which produces the same result as the `man -f` command:

The ability to create your own script files is a very powerful feature of the CLI. If you have a series of commands that you regularly find yourself typing in order to accomplish some task, then you can easily create a Bash shell script to perform these multiple commands by typing just one command:

the name of your script file. You simply need to place these commands into a file and make the file executable.

A common use of the `exec` command is in what is known as wrapper scripts. If the purpose of a script is to simply configure and launch another program, then it is known as a wrapper script.

The options for the `uname` command are summarized below:

-n node name / Network Node Name. -i -- hardware - platform / Hardware platform or unknown.

-m --machine /machine hardware name.

The /bin directory contains executable programs needed for booting a system, commonly used commands, and other programs needed for basic system functionality.
The /sbin directory also contains executable programs; mainly commands and tools designed for system administration.
The `SYNOPSIS` section of a man page can be difficult to understand, but it is a valuable resource since it provides a concise example of how to use the command. For example, consider an example `SYNOPSIS` for the cal command:
The `-f` option to the `man` command will display man pages that match, or partially match, a specific name and provide a brief description of each man page.
Note that on most Linux distributions, the `whatis` command does the same thing as `man -f`. On those distributions, both will produce the same output.

Configuring the Shell

By convention, lowercase characters are used to create local variable names, and uppercase characters are used when naming an environment variable.An environment variable can be created directly by using the `export` command.the `declare` or `typeset` command can be used with the export `-x` option to declare a variable to be an environment variable. the `echo ${PATH}` command would produce the same result as the `echo $PATH` command, the curly braces set the variable apart visually, making it easier to see in scripts in some contexts.
If you create a variable and then no longer want that variable to be defined, use the `unset` command to delete it:Do not unset critical system variables like the `PATH` variable, as this may lead to a malfunctioning environment.
When a new user is created, the files from the `/etc/skel` directory are automatically copied into the new user's home directory. As an administrator, you can modify the files in the `/etc/skel` directory to provide custom features to new users.
The keys that are available for editing a command are determined by the settings of a library called **Readline**. The keys are typically set to match the key assignments found within the `emacs` text editor (a popular Linux editor) by default.
The `!` exclamation mark is a special character to the Bash shell to indicate the execution of a command within the history list.

Absolute paths always start with the `/` character representing the root directory.Using a relative path to execute a file in the current directory requires the use of the `.` character, which symbolizes the current directory:`./my.sh`
The tilde `~` character represents the user's home directory.File names preceded by a period `.` character indicates hidden files. You can view these files in your home directory by using the `ls` command with the all `-a` option.
An environment variable is created with the `export` command: `export JOB=engineer`
To display the value of the variable, use a dollar sign `$` character followed by the variable name as an argument to the `echo` command.

Variables can also be enclosed in the curly brace `{}` characters in order to delimit them from surrounding text. While the `echo ${PATH}` command would produce the same result as the `echo $PATH` command, the curly braces set the variable apart visually, making it easier to see in scripts in some contexts.

Warning
Do not unset critical system variables like the `PATH` variable, as this may lead to a malfunctioning environment.

An environment variable can be created directly by using the `export` command: `export variable=value`

To display the value of the variable, use a dollar sign `$` character followed by the variable name as an argument to the `echo` command. Recall that the `echo` command is used to display output in the terminal.

The `set` command by itself will display all variables (local and environment).To only display environment variables, execute the `env` command.If it is no longer necessary for a variable to be defined, the unset command can be used to delete it:

/home/sysadmin/bin	A directory for the current sysadmin user to place programs. Typically used by users who create their own scripts.
/usr/local/sbin	Normally empty, but may have administrative commands that have been compiled from local sources.
/usr/local/bin	Normally empty, but may have commands that have been compiled from local sources.
/usr/sbin	Contains the majority of the administrative command files.

/usr/bin	Contains the majority of the commands that are available for regular users to execute.
/sbin	Contains the essential administrative commands.
/bin	Contains the most fundamental commands that are essential for the operating system to function.

Recall that when you execute a command without specifying a complete path name, the shell uses the value of the `PATH` variable to determine where the command is stored.

Using the `which` command, you can determine where a command resides in the `PATH`. Execute the following to determine where the `ls` command resides.You can also use the `type` command, which more clearly indicates that the command is not located in the

There are two types of paths commonly used when invoking a command or locating a file in the Linux filesystem: absolute paths and relative paths. Absolute paths always start with the `/` character representing the root directory.
If a command or file seems to be missing, often the reason is that it is not located within directories in the `PATH` variable, and the user is trying to access it with a relative path. While specifying an absolute path should remedy the problem, it can be cumbersome, especially when a file is located several layers deep into a directory structure.

Use the `less` pager command to view your account's `.bashrc` file to see what customizations are set by this file. To exit the `less` command and return to the prompt, press **Q**.
Adding custom paths to the `PATH` environment variable provides users with a means to customize their shell. For example, if you wanted to create a custom shell script that could run from a directory other than where it was located, you could add the path to it by modifying the `PATH` environment variable. The `grep` command can be used to specifically see any changes made to the `PATH` variable in the `.bashrc` file : `grep PATH .bashrc`
The `grep` command provides the ability to filter text and will only display lines that contain the argument passed to the command, in this case, `PATH`.
Execute the `history` command to view previously executed commands:Execute the `history` command again by executing the command history shortcut `!!`

Basic File Management

The `cat` command (derived from the word concatenate) accepts multiple files as input and outputs a merged file.To concatenate the contents of files `/etc/hosts` and `/etc/hostname`, execute the following command:`cat /etc/hosts /etc/hostname.`

The standard output of the `cat` command can be sent to another file by using redirection. Standard output redirection is achieved by following a command with the greater-than `>`character and a destination file. `cat /etc/hosts /etc/hostname > result`

To view the `result` file along with line numbers prefixed to each line, execute the following command:`cat -n result` The `cat` command can be used for viewing text files only.

The `less` command provides a very advanced paging capability. It is usually the default pager used by commands like the `man` command.While you are in the output of the `less` command, you can view the help screen by pressing the **h** key:you can return to the file by pressing the **q** key.To start a search to look forward from your current position, use the slash */* key. Then, type the text or pattern to match and press the **Enter** key.
The `split` command is used to split the input file into two or more files. The syntax for the `split` command is:The `split` command allows files to have a numeric suffix instead of a default alphabetic suffix. To split the `words` file into files prefixed with the name `result` and suffixed by numbers, execute the following commands:`split words -d result.`By default, the `split` command will break apart a file into 1,000-line chunks. The first 1,000 lines of the original file will go into the first file, the second 1,000 lines will go into the second file, etc. The `-l` option can be used to change the default number of lines to split upon. To split the `words` file at every 500th line, execute the following command:`split words result -l 500`
The `nl` command will number the lines of its output. The syntax for the `nl` command is:`~/Documents$ nl newhome.txt.`By default, the `nl` command is used to number non-blank lines in the output. To count the number of lines in the `newhome.txt` file, including blank lines, execute the following command:`nl -ba newhome.txt`

The purpose of the `head` command is to view the beginning of a file or output.`head /usr/share/dict/words.`
If the number of lines is not specified, the `head` command will display the first 10 lines. To view the first 15 lines of the output of the `/usr/share/dict/words` file, execute the following command: `head -15 /usr/share/dict/words.` To view the first 5 lines of the `/usr/share/dict/words` file, execute the following command:`head -n 5 /usr/share/dict/words.`To view the first 20 lines of the output of the `man ls` command, execute the following command:`man ls | head -20`

The `tail` command displays contents from the end of the file and not the beginning.To view the last 10 lines of the `/usr/share/dict/words` file, execute the following command `tail /usr/share/dict/words.` o view the last 5 lines of the `/usr/share/dict/words` file, execute the following command;`tail -5 /usr/share/dict/words.`To view the last 5 lines of the output of the `ls -ltr` command, execute the following command:`ls -ltr /etc | tail -n 5.`To view the contents of the `/etc/hosts` file starting from the 3rd line to the end of the file, execute the following command:`tail -n +3 /etc/hosts.`To display the last 4 lines of the `/etc/hosts` file (the total number of lines in this file is 7, only the last 4 lines will be displayed), execute the following command:`tail -n -4 /etc/hosts.`

The `paste` command will merge the lines of one or more files, line by line, separating them with a tab as a delimiter (separator) by default. The `paste` command is especially useful for files that

contain data in column format. The syntax for the `paste` command is:o merge the two files again using the colon `:` character as the delimiter instead of the default tab delimiter, execute the following command: `paste -d : head tail > total.`

The `cut` command is used to extract fields from a text file. The space and tab are the default delimiters and can be changed using the `-d` option.`cut -d: -f1,3,4 /etc/passwd | head -n 4.` A range of fields can also be provided to the `cut` command. To extract the fields 1 – 4 from the `/etc/passwd` file, execute the following command:`cut -d: -f1-4 /etc/passwd | head -n 4`

Using the `cut` command, a specific field or set of fields can also be extracted from data containing fixed width columns. To extract the contents from column 31 to 43 from the output of the `ls -l` command, execute the following command:`ls -l /etc | head | cut -c31-43`

The `sort` command is useful for working with data organized in columns. It is used to display a file sorted on a specific field of data. `-k` option has one argument: the `2`, which indicates the second field to sort. By default, `sort` breaks up each line of a file into fields using whitespace (tabs or spaces) as delimiters. To specify an alternate delimiter, use the `-t` option.To sort the `/etc/passwd` file using the first field (i.e., the user name: `bin:x:2:2:bin:/bin:/usr/sbin/nologin`) as a key and using the colon `:` character as the delimiter, execute the following command:`sort -t: -k1 /etc/passwd | head -n 4.` To reverse the sorting order in the example from the previous step from ascending to descending, execute the following command: In the command above, the `-k` option has two arguments: the `1` indicates the first field to sort and the `r` argument to reverse the sort.For certain fields, the sorting order required may be numerical instead of alphabetical. In order to have the `sort` command treat a field numerically, add an `n` as an argument to the `-k`option for that key field specification. To treat the sorting field numerically for the third field of the `/etc/passwd` file, execute the following command:`sort -t: -k3n /etc/passwd | head -n 4.` Use the `-u` option to the `sort` command to remove duplicate entries `sort -u output.` o remove duplicate lines from the output file and display the count of duplicates, execute the following command:`sort output | uniq -c.` n the command above, the `-u` option to the `sort` command rearranges the rows in the file and then removes duplicates. The `uniq` command also provides duplicate removal functionality in a slightly different way. It removes duplicates which are already on consecutive lines i.e. it does not sort data.

The `tr` command can be used to translate from one set of characters to another. The `tr` command can be used to translate from one set of characters to another. `cat /etc/hosts | tr 'a-z' 'A-Z.` To replace the five letters (h, i, p, o, and n) with the five characters (!, @, #, $, and %) in the /etc/hosts file, execute the following command: `cat /etc/hosts | tr 'hipon' '!@#$%'.`

The `sed` command is a non-interactive editor that can be used to modify, insert, and delete text based on pattern matching with the specified string. The `sed` command combined with regular expressions gives very powerful text processing capabilities. When performing a search and replace operation, the `sed` command will only replace the first occurrence of the search pattern by default. To replace all occurrences of the pattern, add the global modifier `g` after the final slash.`s/PATTERN/REPLACEMENT/g`.To replace all occurrences of the word `host` with the word `NAME` in the `/etc/hosts` file globally, execute the following command: `cat /etc/hosts sed 's/host/NAME/g' /etc/hosts.` The `sed` command is also able to insert text before a pattern. For this type of expression, do not use the `s` character before the first slash; use an insert change with the `i\` expression. `/PATTERN/i\TEXT\` To insert the word `Report` in the /etc/hosts file preceding the line containing `127.0.0.1`, execute the following command: `sed '/127.0.0.1/i\Report' /etc/hosts.` The `sed` command is also able to insert text after a pattern. For this type of expression, use an insert change with the `a\` expression. `/PATTERN/a\TEXT\.` To append the string `End of report` in the `/etc/hosts` file following the line containing `allrouters`, execute the following command: `sed '/allrouters/a\End of report' /etc/hosts .` The `sed` command can also be used to search for a line of text containing a pattern and then delete the lines that match. Place the pattern to search for between two slashes followed by a `d` to carry out this operation:To delete lines that contain "localhost" in the `/etc/hosts` file, execute the following command: `sed '/localhost/d' /etc/hosts .`

Regular Expressions

The `grep` command is used to demonstrate the use of regular expressions.There are two types of regular expressions – basic and extended. While the basic expressions are interpreted by most commands, the extended expressions can be used along with an option in commands that support their interpretation.
An empty string is a string that has nothing in it. It is commonly referred to as "".
The question mark `?` characters in a string will match exactly one character.
When the hyphen `-` character is used with square brackets, it is referred to as a range.
Normally, you want to avoid using special characters like `*, ?, [` and `]` within file names because these characters, when used in a file name on the command line, can end up matching other file names unintentionally.
If the first character inside of the square brackets is either an exclamation `!` character or a caret `^` character, then that first character has a special meaning of not the following character.

Period operator	.	Matches any one single character.

List operator	[ ] [^]	Defines a list or range of literal characters that can match one character. If the first character is the negation ^ operator, it matches any character that is not in the list.
Asterisk operator	*	Matches zero or more instances of the previous character.
Front anchor operator	^	If ^ is the first character in the pattern, then the entire pattern must be present at the beginning of the line to match. If ^ is not the first character, then it is treated as an ordinary literal ^ character.
Back anchor operator	$	If $ is the last character in the pattern, then the pattern must be at the end of the line to match, otherwise, it is treated as a literal $ character.

To find the occurrences of the pattern root in the /etc/passwd file, execute the following command:

The pattern argument of a command should be protected by strong quotes to prevent the shell from misinterpreting them as special shell characters. This means that you should place single quotes ‘ around a regular expression.

In basic regular expressions, putting a backslash `\` character in front of another character means to match that character literally. For example, using the `\.` pattern is an appropriate way to match the `.`character. To display the file names of the files in the `/etc` directory that start with the letters `rc`, followed by a digit in the range `3-6`, and ending in the file extension `.d`, execute the following command with the regular expression `rc[3-6]*\.d`: `ls /etc | grep 'rc[3-6]*\.d'`

The `grep -E` command is used to recognize the extended regular expression character. Extended regular expression patterns support the basic regex operators PLUS the following additional operators:

Extended Regex	Operators	Meaning
Grouping operator	( )	Groups characters together to form a subpattern.

Asterisk operator	*	Previous character (or subpattern) is present zero or more times.
Plus operator	+	Previous character (or subpattern) is present at least one or more times.
Question mark operator	?	Previous character (or subpattern) is present zero or one time (but not more).
Curly brace operator	{,}	Specify minimum, maximum, or exact matches of the previous character (or subpattern).
Alternation operator	|	Logical OR of choices. For example, abc|def|xyz matches abcor def or xyz.

`tail /etc/passwd | grep -E ':[0-9]+:'`

the egrep command is an alternative to the `grep -E` command. To match the same extended regular expression.`ls /etc | egrep 'rc[3-6]+\.d'`

The extended regex curly brace {} operator is used to specify the number of occurrences of the preceding character or subpattern.

Pattern	Meaning
a{0,}	Zero or more a characters
a{1,}	One or more a characters
a{0,1}	Zero or one a characters
a{5}	Five a characters
a{,5}	Five or fewer a characters
a{3,5}	From three to five a characters

Special characters, such as the asterisk `*` operator and the plus `+` operator, need to be escaped using the backslash `\` character to avoid their interpretation as a special character during expression evaluation; `grep "\*" /etc/rsyslog.conf`

An alternative to using the backslash `\` character to escape every special character is to use the `fgrep` command, which always treats its pattern as literal characters. `fgrep "*" /etc/rsyslog.conf`

Regular expressions also use the backslash `\` character for designated backslash character combinations, called backslash sequences. Backslash sequences can represent special operators or character classes. For example, `\b` indicates the word boundary operator, `\s` indicates the whitespace character, and `\w` indicates a word character.`sed 's/\bis\b/was/' /etc/wgetrc | head -n -45 | tail -n -15`

The `grep` command provides the `-i` option for making the pattern matching case insensitive.

`grep -i "abid" /usr/share/dict/words`

The `-v` option for the `grep` command will cause all lines that don't match the pattern to be displayed. `grep -v "local" /etc/hosts.`

the `grep` command has been used for searching patterns within a single file. The `grep` command can also be used for searching in multiple files.`grep "test" /etc/m*`

To search multiple files but view only the file names instead of every matching line, execute the following command:`grep -l "script" /etc/mime*`

The vi Editor

The `vi` editor works in three modes: command mode, insert mode, and ex mode

Escape key to return to the command mode. Type `dd` to delete the current line Move to the end of the document by typing a `G` character:`G.` To add a new line at the end of the document, type an `o` character and then type `the end.` Go to line #1 by typing `1G.`Delete the current word by typing `dw.`Type `1G` to return to the first line and then `5dd` to delete five lines.Copy three lines by typing `3yy.` Then, move to the end of the document by typing `G`. Paste the three lines by typing `p.`Undo the paste command by typing the letter `u.` Type `:q` and then press the **Enter** key to attempt to quit the document.To quit without saving, type `:q!`and press the **Enter** key to quit the document and not save any. Type `1G` to return to the first line of the file (if needed). Change (replace) the first line of the document by typing `cc` and then. Press the **Escape** key to return to the command mode. Then, type `:w` to save the changes

Standard Text Streams and Redirection

One of the key points in the UNIX philosophy was that all CLI commands should accept text as input and produce text as output. As this concept was applied to the development of UNIX (and later Linux), commands were developed to accept text as input, perform some kind of operation on the text and then produce text as output. Commands that read in text as input, alter that text in some way, and then produce text as output are sometimes known as filters.

`COMMAND > FILE`
 	
`COMMAND 1> FILE`
	Create or overwrite `FILE` with the standard output of `COMMAND`

`COMMAND >> FILE`
 	
`COMMAND 1>> FILE`
	Create or append to `FILE` with the standard output of `COMMAND`

`COMMAND 2> FILE`	Create or overwrite `FILE` with the standard error of `COMMAND`
`COMMAND 2>> FILE`	Create or append to `FILE` with the standard error of `COMMAND`

The `/dev/null` file is like a trash can, where anything sent to it disappears from the system; it's sometimes called the bit bucket or black hole. Any type of output can be redirected to the `/dev/null` file; most commonly users will redirect standard error to this file, rather than standard output.

`COMMAND &> FILE` **`COMMAND > FILE 2>&1`**	**Create or overwrite `FILE` with all output (`stdout`, `stderr`) of `COMMAND`**
`COMMAND &>> FILE` **`COMMAND >> FILE 2>&1`**	**Create or append to `FILE` with all output (`stdout`, `stderr`) of `COMMAND`**

One common way that text files are used as standard input for commands is by creating script files. Scripts are plain text files which are interpreted by the shell when given the proper permissions and prefaced with `#!/bin/sh` on the first line, which tells the shell to interpret the script as standard input:

```
#!/bin/sh  echo HelloWorld
```

When the script file is invoked at the prompt using the `./` syntax, the shell will run all commands in the script file and return the result to the terminal window, or wherever the output is specified to be sent to:

```
./examplescriptfile.sh
```

In some cases, it is useful to redirect standard input, so it comes from a file instead of the keyboard. A good example of when input redirection is desirable involves the `tr` command. The `tr` command translates characters by reading data from standard input; translating one set of characters to another set of characters and then writing the changed text to standard output.The `tr` command won't accept a file name as an argument on the command line. To perform a translation using a file as input, utilize input redirection. To use input redirection, type the command with its options and arguments followed by the less-than < character and a path to a file to use for input.

Important

Do not attempt to use the same file for input and output redirection, as the results are probably not desirable (you end up losing all data). Instead, capture the output and place it into another file; use a different file name as shown below:

```
sysadmin@localhost:~$ tr 'a-z' 'A-Z' < Documents/animals.txt > animals.ne
```

The following command line will extract some fields from the `os.csv` file with the `cut` command, then sort these lines with the `sort` command, and finally eliminate duplicate lines with the `uniq` command:

```
cut -f1 -d',' Documents/os.csv | sort -n | uniq
```

A server administrator works like a plumber, using pipes, and the occasional `tee` command. The `tee` command splits the output of a command into two streams: one directed to standard output, which displays in the terminal, and the other into a file.The `tee` command can be very useful to create a log of a command or a script. For instance, to record the runtime of a process, start with the `date` command and make a copy of the output into the `timer.txt` file:

```
date | tee timer.txt
```

The `sleep` command is being substituted for a timed process; it pauses for a given number of seconds:

```
sleep 15
```

Then run the `date` command again. This time, append the time to the end of the `timer.txt` file by using the `-a` option:`date | tee -a timer.txt` To run all of the above commands as a single command, use a semicolon `;` character as a separator:`date | tee timer.txt; sleep 15; date | tee -a timer.txt`

A command's options and parameters are usually specified on the command line, as command line arguments. Alternatively, we can use the `xargs` command to gather arguments from another input source (such as a file or standard input) and then pass those arguments to a command. The `xargs` command can be called directly and will accept any input:

Important

Pressing **Ctrl**+**D** after exiting the `xargs` command by using **Ctrl**+**C** will log you out of the current shell. To send the input of the `xargs` command to the `echo` command without logging out of the shell, press **Ctrl**+**D** while you are still running the `xargs` command.

The `xargs` command is most useful when it is called in a pipe. In the next example, four files will be created using the `touch` command. The files will be named `1a`, `1b`, `1c`, and `1d` based on the output of the `echo` command.`echo '1a 1b 1c 1d' | xargs touch.`These four files can be removed just as easily by changing the `touch` command to the `rm` command:`echo '1a 1b 1c 1d' | xargs rm`

A delimiter can be set using the `-d` option with the `xargs` command. To view the contents of the `~/Documents` directory containing the word `alpha` with all instances of the dash `-` character replaced with a space, type the following:`Documents$ ls | grep alpha | xargs -d '-'`

the output of the `cat` command was used as input to the `touch` command using `xargs` and a command-line pipe. The xargs command can also be used with pipes to send the output of a command as an argument to another command:`find ~ -maxdepth 1 -name 'D*' | xargs d`

Managing Processes

To view all of the processes on the system using non-BSD options, execute the following command:`ps -ef.`To see all processes (processes belonging to all users and not limited to the current shell) as well as display the user owners of the processes, using BSD options, execute the following command:`ps aux.`The `watch` command can be used to monitor recurring processes by using the following syntax:`watch ps.`Press **Ctrl**+**C** to stop the `watch` command:By default, the `watch` command executes commands every two seconds. To change the interval at which the

`watch` command will execute commands, use the `-n` option, followed by the specific interval desired.`watch -n 15 tail /var/log/ndg/web.log.` The `watch` command can be used with the `-d` option to highlight the differences in the successive updates of a command. To monitor the `/var/log/ndg/web.log` file, update the display of the log file's contents in five seconds increments, and highlight changes between each update, execute the following command:`watch -n 5 -d tail /var/log/ndg/web.log`
o see all processes (processes belonging to all users and not limited to the current shell) as well as display the user owners of the processes, use the `aux` BSD option:

When one process starts another, the first process is referred to as the parent process, and the new process is called a child process. So, another way of thinking of a foreground process is that when running in the foreground, a child process doesn't allow any further commands to be executed in the parent process until the child process ends.
When a command may take some time to execute, it may be better to have that command execute in the background. When executed in the background, a child process releases control back to the parent process (the shell, in this case) immediately, allowing the user to execute other commands.
The job number of a process is sometimes followed by a minus - or a plus + character. The plus + character denotes the last process that was started, while the minus - character denotes a process started prior to the latest one.
While there are still background processes being run in the terminal, they can be displayed by using the `jobs` command.
A command that has been paused or sent to the background can then be returned to the foreground using the `fg` command.To put the process in the background again, use **Ctrl+Z** to stop the `./text.sh` process, then execute the `bg` command:
A signal is a message that is sent to a process to tell the process to take some sort of action, such as stop, restart, or pause. There are several commands that will allow you to specify a signal to send to a process; the `kill` command is the most commonly used. The syntax for the `kill` command looks like the following:When sending a signal, specify one or more processes to send the signal to. There are numerous techniques to specify the process or processes. The more common techniques include:

- Specifying the process identifier (PID)
- Using the `%` (percent sign) prefix to the job number

Execute the `./test.sh` script file in the background, then execute the following command to stop the background process:The `killall` command can also be used to terminate one or more processes. The following demonstrates syntax that can be used for the `killall` command.`sleep` processes and then stop them all with a single command, as shown below:`sleep 100 &`

A user can influence the priority that will be assigned to a process by setting a value of something called niceness.To set the initial niceness of a command, use the `nice` command as a prefix to the command to execute. The `-n`option indicates the desired niceness value.

ou can use the `renice` command to change an existing process priority. Enter the commands below, using the PID assigned to your sleep process.`renice -n 15 -p PID`.se the `renice` command to adjust the priority back to normal. Like the `nice` command, the `-n` option indicates the niceness value. The `-p` option indicates the process ID to operate on.

`Uptime` command to display basic process information, including the load average of the system:

Execute the following command to display basic system memory statistics:`free`
To display a real-time view of running processes, execute the following command:`top`

Multi-session command line utilities allow users to manage multiple processes inside a single Bash shell environment. The `screen` and `tmux` commands allow the user to start processes in a session that can be detached while still running, then re-attached and managed by various methods.

The `screen` command allows for multiple processes to run within separate sessions under a single terminal. Start a `screen` session by executing the following command:`screen`

A useful feature of the `screen` command is the ability to detach a session, then reattach it later. To attach and detach a `screen` session, you will need to use the `screen` command keys. All `screen` commands start with a prefix key, the keystrokes **Ctrl**+**A**, followed by a command key to make an action happen.

Press the prefix key **Ctrl**+**A** and then the detach **D** command key, which detaches the current `screen` session and returns the user to the shell prompt.o list currently running `screen` sessions, run the `screen` command with the list `-l` option:`screen -list`

You can now re-attach the session by using the resume `-r` option with either the PID of the session or by the name of the session:`screen -r PID`.To exit the `screen` command, use the `exit` command.
The `tmux` command, short for terminal multiplexer, allows for multiple terminals to be opened and viewed on the same screen . In this step, you will start a `tmux` session, using the `new-session` option, and run the `top` command in the current `tmux` session:`tmux new-session 'top'`
Detach from the current `tmux` session, which is running the `top` command, by pressing the **Ctrl**+**B** key sequence and then the **D** key to return to the shell prompt again:`CTRL+b then d`
To re-attach the running `tmux` session `0`, use the `tmux attach` command with the target-session `-t` flag:`tmux attach -t 0`.To enable a new terminal in the `tmux` session running in a side-by-side vertical window, press **Ctrl**+**B** then **%** (**Ctrl** and **b**, then **Shift**+**5**):

Archive Command

The `gzip` and `gunzip` commands are used to compress and uncompress a file, respectively. The `gzip` command replaces the original file with the compressed `.gz` file.
The `gzip` command should be used with caution since its default behavior is to replace the original file specified with a compressed version.
The `gunzip` command reverses the action of `gzip`, so the `.gz` file is uncompressed and replaced by the original file.Use the `-l` option with `gunzip` to list the amount of compression of an existing file and then use the `gunzip` command alone to decompress,To retain the original file while compressing using the `gzip` command, use the `-c` option `gzip -c animals.txt > animals.txt.gz`.

he `zcat` command is used to display the contents of a compressed file without actually uncompressing it.`zcat animals.txt.gz`
The `gzip` and `gunzip` commands support recursion with the `-r`option. In order to be able to compress files with the `gzip` command recursively, a user needs to have the correct permissions on the directories the files are in. Typically, this is limited to directories within the user's own home directory.
To avoid having to repeatedly type the same file or directory name, type the first few characters of the file name and press the **Tab** key. Alternatively, you can use the **Esc+.** (the **Escape Key** and the period **.** character) shortcut to recall the last file name used.

Permissions can have an impact on file management commands, such as the `gzip` and `gunzip` commands. To `gzip` or `gunzip` a file within a directory, a user must have the write and execute permission on a directory as well as the readpermission on the file.

The `bzip2` and `bunzip2` commands work in a nearly identical fashion to the `gzip` and `gunzip` commands:
Similar to the `gzip` and `gunzip` commands, the `bzip2` and `bunzip2` commands are also used to compress and uncompress a file. The compression algorithm used by both commands is different, but the usage is very similar. The extension of the files created by `bzip2` command is `.bz2`.
While the `gzip` command supports recursion with the `-r` option, the `bzip2` command does not support a separate option for recursion. So, `bzip2` cannot be used to compress a nested directory structure.

The `xzcat` command is used to print the contents of files compressed with the `xz` command to standard output on the terminal without uncompressing the target file.the `unxz` command to uncompress the `longfile.txt.xz` file:`unxz longfile.txt.xz`

The `tar` command is typically used to make archives within Linux. The `tar` command provides three main functions: creating, viewing, and extracting archives:

- **Create**: Make a new archive out of a series of files.
- **Extract**: Pull one or more files out of an archive.
- **List**: Show the contents of the archive without extracting.

`-c`	Create an archive.
`-f ARCHIVE`	Use the `ARCHIVE` file. The argument `ARCHIVE` will be the name of the resulting archive file.

Use the `-t` option to the `tar` command to view a list (table of contents) of a tar file.`tar -tf vim.tar`

The verbose `-v` option can be used with the `tar` command to view the table of contents of the archive. To view the detailed listing of the contents of the `vim.tar` file, execute the following command:`tar -tvf vim.tar`

To extract the files from the tar file, use the `-x` option.

`-x`	Extract files from an archive.
`-f ARCHIVE`	Operate on the given archive.

To extract the files from the `vim.tar` into another directory, use the `-C` option to the `tar`command. For example, execute the following commands:`tar -xvf vim.tar -C /tmp`

Archiving files is an efficient way of making backups and transferring large files. The most commonly used compression utilities are `zip` and `unzip`. The `zip` command is very useful for creating archives that can easily be shared across multiple operating systems.The `-r` option allows the `zip` command to compress multiple directories into a single file recursively.`zip -r myperl.zip /etc/perl`

To view the contents of a zip file without unpacking it, use the `unzip` command with the list `-l` option

`unzip -l myperl.zip`

The `unzip` command is used to extract the files from the zip archive file.

The `cpio` command is another archival command, which can merge multiple files into a single file.

The `cpio` command works with the original POSIX specification and should be available on all Linux and Unix systems. It is considered a legacy application, and although administrators need to be aware of it, most systems provide better alternatives for archiving directories.

The `cpio` command operates in two modes: copy-in mode and copy-out mode. The copy-out mode is used to create a new archive file. The `-o` option puts the `cpio` command into copy-out mode. The files can be provided via standard input or redirected from the output of another command to produce a file stream which will be archived.To archive all the `*.conf` files in the current directory, use the following command:`cp /etc/*.conf .    ls *.conf | cpio -ov > conf.cpio` the verbose `-v` option is used to list the files that the `cpio` command processes

The copy-in mode is used to extract files from a cpio archive file. The `-i` option enables copy-in mode. In addition, the `-u` option can be used to overwrite existing files and the `-d`option is used to indicated that directories should be created. To extract the files from the `conf.cpio` file into the current directory, first delete the original files and then use the cat command to send the data into the `cpio` command:`cat conf.cpio | cpio -iud`

The `dd` command is a utility for copying files or entire partitions at the bit level. It can be used to clone or delete entire disks or partitions, creating large "empty" files to be used as swap files and copy raw data to removable devices. The `dd` command uses special arguments to specify how it will work. The following illustrates some of the more commonly used arguments:

`if=FILE`	The input file to be read.
`of=FILE`	The output file to be written.
`bs=SIZE`	The block size to be used. By default, the value is considered to be in bytes. Use the following suffixes to specify other units: `K`, `M`, `G`, and `T`for kilobytes, megabytes, gigabytes, and terabytes.
`count=NUMBER`	The number of blocks to read from the input file.

To create a file named `/tmp/swapex` with 500 "one megabyte" size blocks of zeroes, execute the following command:`dd if=/dev/zero of=/tmp/swapex bs=1M count=500`

File Permissions

The first character of this output indicates the type of a file. Recall if the first character is a dash `-` character, as it is in the output above, this is a regular file. If the character was a letter `d`, it would be a directory.`-rw-r----- 1 root shadow 968 Apr 17 22:31 /etc/shadow`
After the file type character, the permissions are displayed.`-rw-r----- 1 root shadow 968 Apr 17 22:31 /etc/shadow`
After the link count, the file's user owner is displayed.`-rw-r----- 1 root shadow 968 Apr 17 22:31 /etc/shadow`
After the user owner field, the file group owner is displayed.`-rw-r----- 1 root shadow 968 Apr 17 22:31 /etc/shadow`.Note that the user owner of the `/etc/shadow` file is the `root` user and that the group owner is the `shadow` group. The user owner has read and write permission (`rw-`), the group owner only has the read permission (`r--`), and regular users have no permissions (`---`).
Be aware that both instances in the command above use a lowercase "L", not a number one.`ls -l /srv/lab.txt`

The `chmod` command is used to change the permissions of a file or directory. Only the `root` user or the user who owns the file is able to change the permissions of a file.
There are two techniques of changing permissions with the `chmod` command: symbolicand octal. To use the symbolic method of `chmod`, use the symbols summarized in the table below:

Group Symbol	Operation Symbol	Permission Symbol
`u` (user owner)	`+` (add the permission)	`r` (read)

`g` (group owner)	`=` (specify the exact permission)	`w` (write)
`o` (others)	`-` (remove the permission)	`x` (execute)
`a` (all)		

Execute the following command to provide members of the others permission set the ability to both view and modify the `/srv/lab.txt` file:`chmod o+rw /srv/lab.txt`
Recall that you would not normally provide others with more permissions than provided to the group members.
Changing permissions using the octal method requires that the permissions for all three sets be specified. It is based on the octal numbering system in which each permission type is assigned a numeric value:

Octal Value	Permission
4	read
2	write
1	execute
0	none

By adding together the combination of numbers from 0 to 7, any possible combination of read, write, and execute permissions can be specified for a single permission group set. For example, for read, write, and execute: add 4 + 2 + 1 to get 7. Or, for read, not write, and execute: add 4 + 0 + 1 to get 5.

When the setuid permission is set on an executable binary file (a program), the binary file is run as the owner of the file, not as the user who executed it.
`-rwsr-xr-x 1 root root 76496 Jan 25  2018 /usr/bin/chfn`

Recall that the `s` character in the owner's permission set means that this is a setuid file. When executed, this program can access files as if the program was run as the `root` user (the owner of the file). This special permission allows users to change the information in the `/etc/passwd` file. Typically, special permissions are only set by the administrator (the root user), and they perform very specialized functions. They can be set using the `chmod` command, using either the symbolic or octal method.

The setgid permission is similar to setuid, but it makes use of the group owner permissions. In the next few steps, we will demonstrate the use of setgid directories. Start by executing the following command to switch to the `root` account. When prompted, provide the root password:

It is possible for the user owner of a file or directory to change the group owner of that same file/directory by using the `chgrp` command. Verify that the `team` group exists and then change the group ownership of the `/srv/test` directory to the `team` group by executing the following commands:`chgrp team /srv/test`.The `chgrp` command was used above to change the group owner of the `/srv/test`directory from the default primary group `root` to the group `team`.

Add the sticky bit permission to the `/pub` directory so users can only delete the files that they own in this directory, then verify the permissions with the following commands:`chmod o+t /pub`

It may be confusing that there is a fourth value in the output of the previous `umask`command. This is because there are technically four sets of permissions: user owner, group owner, others, and special permissions (the first `0` in the output above). Since special permissions are never set by default, the initial 0 is not necessary when setting the umask value

To set a umask value for new files that would result in default permissions for the owner, remove write permissions for the group owner and remove all permissions from others, the umask value would be `026`:

File Default	`666`	`rw-rw-rw-`
Umask	`-026`	`----w-rw-`
Result	`640`	`rw-r-----`

Remember that a umask value only affects new files and directories. Any existing file will never be affected by the umask value.

The umask value is designed to make it easy for you to specify default permissions for new files and directories. By choosing a good umask value, you save yourself a lot of effort in the future since you won't have to change permissions on new files and directories very often.

Recall that umask values, when set in the shell, are not permanent. To make a umask that will apply to every shell that you open, add the umask command line to your `~/.bashrc` file.

Filesystem Links

Execute the following command to see a soft link file:`ls -l /etc/rc.local`

Soft links to directories are tricky. If you just refer to the soft link (like the first `ls` command above), only the soft link itself will be displayed. If you add a trailing `/` character to the end of the soft link name, then it follows the soft link and displays the contents of the directory that the soft link is linked to.

Hard links share the same inode table. Execute the following command to create a hard link file:An inode table contains the metadata for the file. This is all the information about the file besides the file name. When two files share an inode table, they are essentially the same file, but with different names.Note that the hard linked files are identical, besides their names. If you add another hard link, the hard link count increases. Execute the following

You can tell that these files are all hard linked together by using the `-i`option to the `ls` command:`ls -i`.Because all of these files share the same inode number, they are hard linked together.Unfortunately, hard links can't be made to directories.

Hard Link Advantages

- Hard linked files are indistinguishable by programs from regular files.
- If files are hard linked, then they are always contained within one filesystem.
- Hard links don’t have a single point of failure.
  Once files are hard linked together, there is no concept of the original. Each file is equal, and if one link is deleted, the others still work, you don't lose the data. As long as one hard link remains, the file data can be accessed.
  This is unlike soft links in which the data is stored in the file that is being pointed to, meaning that if the original file is removed, all of the soft links are now pointing to nothing. Consider the following example in which access to the data fails if the original file is deleted. The `mytest.txt` file is a symbolic link to the `text.txt` file:

Soft Link Advantages

Soft links can be made to a directory file; hard links cannot.
Another limitation of hard links is that they cannot be created on directories. The reason for this limitation is that the operating system itself uses hard links to define the hierarchy of the directory structure. The following example shows the error message that is displayed if you attempt to hard link to a directory:

```
sysadmin@localhost:~/Documents$ cd

sysadmin@localhost:~$

sysadmin@localhost:~$ ln /bin binary

ln: `/bin': hard link not allowed for directory
```

Linking to directories using a symbolic link is permitted:

```
sysadmin@localhost:~$ ln -s /bin binary
```

```
sysadmin@localhost:~$ ls -l binary
```

```
lrwxrwxrwx 1 sysadmin sysadmin 4 May  9 04:04 binary -> /bin
```

-
- Soft links can link to any file.
  Soft links can be made from a file on one filesystem to a file on another filesystem; hard links cannot. Since each filesystem (partition) has a separate set of inodes, hard links cannot be created that attempt to cross file systems:

In general, if you need to link to a file on another filesystem or to a directory, then soft links are the correct type to use. Otherwise, you should make use of hard links.

Hardware Configuration

For a more detailed look at your CPU's features, execute the following command:`cat /proc/cpuinfo`Note the highlighted `flags:` field in the output above. One of the key settings of the `/proc/cpuinfo` file is the flags that the CPU supports, which are a feature of the CPU. Some advanced CPU functions require specific flags.The `df -h` command can be used to determine which type of drive is being used in a Linux based computer.

The `free` command gives details on total, used, and free memory the system has access to, including swap space on fixed disks that can be used as temporary storage for memory operations. Your output may differ, depending on system loads.To display information about memory usage, execute the following command:`free`

The `/proc/meminfo` file provides a very detailed breakdown of how much memory a system has and how it is being used. For a more detailed look at memory usage, execute the following command;`cat /proc/meminfo | less`

Execute the following command to display the USB devices that are attached to the system:`lsusb`

CentOS image to complete the following steps.

Display the hardware devices by executing the following command:`lspci`

Display devices along with their device code by executing the lspci command with the -nn option:`lspci -nn`.Using the highlighted value from the output of the previous command, display the details about the USB Controller:`lspci -v -d 8086:7020`

Display all of the SATA drives on the system by executing the following command:`ls /dev/sd*`.Devices that begin with `sd` in the `/dev` directory are device files that represent SATA or SCSI devices.

Display kernel modules that are loaded into memory by executing the `lsmod` command:`lsmod | less`

Display information about the `dm_mod` module by executing the following command:`modinfo dm_mod | less`.Display information about the `dm_log` module by executing the following command:`modinfo dm_log`

Determine if the `fat` or `vfat` modules are currently loaded into memory by executing the following command:`lsmod | grep fat`.The lack of output from the previous command indicates that neither the `fat` nor `vfat` module is currently loaded into memory.

Load the `vfat` module into memory by executing the following comma;`modprobe vfat`

Verify that the `vfat` module has been loaded by executing the following command:`lsmod | grep fat`

Try to remove the `fat` module from memory by executing the following command:`modprobe -r fat`

Remove the `vfat` module from memory and verify by executing the following commands `modprobe -r vfat lsmod | grep fat`

Bootloaders

The boot process starts with the bootloader, the program that loads the kernel into memory and executes instructions to boot the system. Typically, administrators do not find it necessary to make many changes to the bootloader, but you should know how to modify some of the key GRUB configurations, as well as know how to interact with the bootloader interactively when the system is booting.

Please use the Ubuntu image to complete the following steps.

This system supports GRUB2. View the first 10 lines of the `/boot/grub/grub.cfg` file by executing the following command:`head /boot/grub/grub.cfg`
modify settings in the `/etc/default/grub` file. To edit this file, first launch the `vi` editor:`vi /etc/default/grub`
The `update-grub` command must be executed after modifying this file. Execute that command, then `reboot` the system and watch the screen for the countdown:`update-grub`.
Press the **Down Arrow** key once so the "...(`recovery mode`)" option is selected and press the **e** key to enter the edit mode:This option will boot the system to a recovery runlevel where an administrator can fix system problems:

CentOS image to complete the remaining steps.
This system supports traditional GRUB. View the `/boot/grub/grub.conf` file by executing the following command:`cat /boot/grub/grub.conf`
The `timeout` value determines how long the user has before the system starts booting to the default OS. In this example, there is only one OS, defined by the `title` directive. The `hiddenmenu`

directive indicates that the GRUB menu is not displayed by default during the countdown provided by the GRUB before booting the system.Modify the `/boot/grub/grub.conf` file by executing the following command:`vi /boot/grub/grub.conf`
Next, change the `timeout` to `60` and "comment out" the `hiddenmenu` directive by executing the following `vi` commands .
Verify your changes by viewing the `/boot/grub/grub.conf` file by executing the following command:`cat /boot/grub/grub.conf`
In traditional GRUB, changes to the configuration file do not require running any commands after editing the file.
The line that you would most likely edit is the `kernel` line. By adding (or removing) arguments on this line, you can affect how the system will boot. Keep in mind that any changes made here are not permanent. To make them permanent, you must edit the `/boot/grub/grub.conf`file after booting the system.
The `rhgb` argument stands for Red Hat Graphical Boot. Instead of displaying text messages during boot, a graphical progress bar is displayed. While more user-friendly, this doesn't help the administrator troubleshoot boot problems.

The `quiet` argument suppresses many of the text messages that are displayed if the `rhgb` argument isn't used. Again, this may be more user-friendly, but an administrator should remove this setting when troubleshooting boot problems.

Press the **Backspace** key 10 times to remove the `rhgb` and `quiet` settings. Then, type `single`:
By adding `single` to the end of the kernel line, the system will be booted to the single user runlevel.

Type the `runlevel` command to verify that you are at runlevel `S`.At the single user runlevel, very few processes are running, and only the administrator can access the system. The purpose of this runlevel is to troubleshoot critical system problems.

Runlevels

The Linux kernel can recognize runlevel values from 0 to 9, typically only run levels 0 through 6 are used.
The `who -r` command also displays the current system runlevel. One benefit of this technique is that it will display the date and time that the current runlevel was reached:
Both the traditional SysVinit and Upstart support passing runlevels to the kernel as parameters from the bootloader to override the default runlevel.
The root user can also change runlevels while the operating system is running by using several commands, including the `init` and `telinit` commands
Changing runlevels will affect the applications or services you are running and can cause data loss or connection interruption for users accessing the system for those services.
For example, to take the system to runlevel 1 execute the `systemctl isolate rescue.target` command. Likewise, to natively go to runlevel 5, execute the `systemctl isolate graphical.target` command.To bring the system down to runlevel zero, execute the `halt`, `poweroff`, or `shutdown` command.

If a system is using the traditional `init` process to manage system services, then the scripts in the `/etc/rc.d/init.d` directory are used to manage the state of those system services. For convenience, this directory will usually have a symbolic link from the `/etc/init.d` file.
Instead of having to type the full path name to the script, many systems provide a `service` script that allows the `init`script to be executed without having to type the full path to the script.The `chkconfig` command can be used to view what services will be started for different runlevels.
To view all the services that are set to start or stop automatically, the administrator can execute the `chkconfig --list` command and the output would look something like the following (although there would be many more lines of output):

Because there are three different types of boot systems, traditional `init`, Upstart and systemd, the logical question is, "Which one does my system use?" The easy answer to this question is to check for the existence of two directories: the `/etc/init` and the `/etc/systemd` directory.

If your system has a `/etc/init` directory, then your system is using Upstart. If your system has a `/etc/systemd` directory, then your system is using systemd. Otherwise, your system is using traditional `init`.

Linux systems use the Advanced Configuration and Power Interface (ACPI) event daemon `acpid` to notify user-space programs of ACPI events

If an administrator were to create an `init` script named `serviced` and store it in the `/etc/rc.d/init.d` directory, the `chkconfig --add SERVICE` command would need to be executed first before using either the `chkconfig SERVICE on` or `chkconfig SERVICE off` command. A command similar to this is executed when a new software package containing a service is installed.
To disable a service without uninstalling it, an override file can be created in the `/etc/init` directory. This file should have the same name as the service configuration file, but ending in `.override` instead of `.conf`. This is the preferred technique over commenting out the "start on" lines.
The `systemctl` command is used in systems that have systemd as a replacement for the traditional `init` process. This one command can be used to manually control the state of services, enable or disable automatic starting of services, as well as change system targets.

List the contents of the `/etc/init.d` directory:`ls /etc/init.d`
Execute the following command to see the current status of the `sshd` service:`/etc/init.d/sshd status`
Execute the following commands to stop the `sshd` service and verify that it is stopped:`/etc/init.d/sshd stop`
Execute the following commands to start the `sshd` service and verify that it is started:`/etc/init.d/sshd start`
`/etc/init.d/sshd status`
Execute the following commands to restart the `sshd` service and verify that it is started:`/etc/init.d/sshd restart /etc/init.d/sshd status`

Execute the following command to determine which arguments can be passed to the `/etc/init.d/sshd` command:`/etc/init.d/sshd`
Use the `more` command to display the contents of the `/etc/init.d/sshd`file:`more /etc/init.d/sshd`

Note that the third line of this file, `Start up the OpenSSH server daemon`, describes what this script does. The `description` line provides more details and the `processname` line provides the exact name of the process that this script starts and stops.

Suppose the description provided isn't really enough for you to understand what this service actually does. To learn more, you could look at the man page for `sshd` (although in this lab environment, the man pages don't exist, so this would be more of a "real world" solution).

Execute the following commands to view a list of the scripts that are started when the system is brought to runlevel 3:`cd /etc/rc.d/rc3. d1s S*`

Execute the following command to view a list of the scripts that are stopped when the system is brought to runlevel 3:`1s K*`

Execute the following command to see the `chkconfig` information for the `sshd` script:`grep chkconfig /etc/init.d/sshd`

Execute the following command to display when the `sshd` is started:`chkconfig --list sshd`

Execute the following command to verify that runlevels 2, 3, 4, and 5 contains start scripts for the `sshd` service:`1s /etc/rc.d/rc[0-6].d/S*sshd`

Execute the following command to verify that runlevels 0, 1, and 6 contain stop scripts for the `sshd` service:`1s /etc/rc.d/rc[0-6].d/K*sshd`

Execute the following commands to have the `sshd` script stopped at all runlevels and confirm:`chkconfig sshd off chkconfig --list sshd`

Execute the following commands to have the `sshd` script started only at runlevel 3 and confirm:`chkconfig --level 3 sshd on chkconfig --list sshd`

Execute the following command to verify the runlevels that contain stop scripts for the `sshd` service:`1s /etc/rc.d/rc[0-6].d/K*sshd`

Execute the following command to display the current runlevel:`runlevel`

Execute the following command to view the default runlevel `tail /etc/inittab`

Use the `vi` command to modify the `/etc.inittab` file in order to change the default runlevel:`vi /etc/inittab`

Execute the following command to verify the status of the `sshd` daemon:`/etc/init.d/sshd status`

Execute the following system to runlevel 3 by executing the following command:`telinit 3`

Wait for the process to comment and verify that the `telinit` command took the system to runlevel 3 by executing the following command:`runlevel`

Verify that the `sshd` daemon started by checking the status:`/etc/init.d/sshd status`

View the contents of the `/etc/init` directory `ls /etc/init`

Recall that the `/etc/init` directory is used by Upstart. If this distribution purely used Upstart (and no traditional init scripts), then there would be more files in this directory to define when services would start.

View the contents of the `/etc/init/control-alt-delete.conf` file:`more /etc/init/control-alt-delete.conf`

Note that you should not modify this file directly. Instead, create a file called `control-alt-delete.override` and specify your customizations in that file.

To modify the behavior of the system when someone performs a control-alt-delete key sequence, first make a copy of the `/etc/init/control-alt-delete.conf` file:`cd /etc/init cp control-alt-delete.conf control-alt-delete.override`

Change the `shutdown` command so it doesn't happen immediately, but rather that it waits 5 minutes to allow everyone the time to save work and log off. First, open the file in the `vi` editor:`vi control-alt-delete.override`

By replacing `now` with a `+5`, the `shutdown` command will wait 5 minutes before rebooting the system.Do not attempt to perform a `control-alt-delete` after finishing the previous step. This was just a demonstration of how to create the file, not something designed to be tested in this virtual machine.

Ubuntu image to complete the following steps.

Execute the following command to view the contents of the `/etc/init directory`:`ls /etc/init.d | more`

Recall that the configuration files in the `/etc/init.d` directory determine which services are started at different run levels. For example, execute the following command to see which runlevels the `ssh` service starts:`grep runlevel /etc/init.d/ssh` Note in the output above, the string `"$runlevel"= S` indicates that the runlevel is single user mode.

While Ubuntu mostly uses Upstart, there are a few traditional init scripts that you can see in the `/etc/init.d` directory: `ls /etc/init.d` Most of these init scripts are for backward compatibility for older services. An example would be the `ssh` service functionality even though it is configured as an Upstart service.

Execute the following command to restart the `ssh` service:`/etc/init.d/ssh restart`

Mounting Filesystems

Use the `fdisk` command to display the current partitions:`fdisk -l`

`Device`	The specific partition that the row is describing. For example, `/dev/sda1` is the first partition on the first SATA hard drive.
`Start`	The starting sector of the partition.
`End`	The ending sector of the partition.
`Blocks`	The size of the partition in blocks.
`Id`	An identifier which is used to tell the kernel what type of filesystem should be placed on this partition. For example, the value `83` indicates that this partition should have an ext2, ext3, or etx4 filesystem type.
`System`	A human-readable name that indicates the type of filesystem the `Id` column refers to. For example, `83` is a Linux filesystem.

In this step, we will begin creating a new partition on the `sdb` drive. Execute the following command to start this process:`fdisk /dev/sdb`
At the `Command` prompt, type the `m` command to display the help section:`m`
At the `Command` prompt, type the `p` command to display the current partition table. The output should be the same as when you previously executed the `fdisk -l` command:`p`
While it is not critical for you to understand all of the output of this command, a brief summary of the more useful output is provided below:

`Disk /dev/sdb`	Disk name (`/dev/sdb`) and size in MB and bytes.
`Units`	Available units are sectors, a fixed unit of user accessible storage, which in this case is 512 bytes, the minimum unit of storage available to the system.

`Sector size (logical/physical)`	The total number of sectors is the most important since partitions are done by sectors. Note that a block and a sector are the same things, in this case.
`I/O size`	This is somewhat useful because if you are trying to figure out how big an existing partition is, you can take the number of sectors and multiply it by the value provided here. So, if a partition is 2,000 sectors in size and there are 512 bytes per sector, then the partition is 1,024,000 bytes in size. An easier way to think of it is that 512 bytes is 0.5 MB. So, if a partition is 2,000 sectors in size, it is 1,000 MB in size (or approximately 1GB in size).

At the Command prompt, type `n` to create a new partition:`n`
At the `Select` prompt, type the `e` command to specify that you are going to create an extended partition. This will allow you to create more partitions, called logical partitions, within the extended partition:`e`
Recall that you can have up to 4 primary partitions. One of those can be an extended partition. Within the extended partition, you can create more partitions called logical partitions. It is critical to understand this because if you use all 4 primary partitions on a hard disk, then you will be unable to create any more partitions, even if there is unpartitioned space available on the hard disk.
At the `Partition number` prompt, type `2` to specify that you are going to use the second partition number. This will result in a partition device name of `/dev/sdb2`:`2`
Note that you never use extended partitions directly. In other words, you won't create filesystems on extended partitions, and you won't mount them. Their sole purpose is to be a container for logical partitions.
The next prompt asks for the block (or sector) this partition will start on. To accept the default value, simply press the **Enter** key:`Enter`
The next prompt, `Last sector…` asks for where the new partition will end. You can either provide the last sector, the number of sectors or a specific size (in units of kilobytes, megabytes, gigabytes, terabytes, or petabytes). The default value is to use the rest of the drive, which normally is what you want to use for extended partitions. Press the **Enter** key to accept the default value:
If you wanted to specify the `Last sector` (think of this as the ending sector), you could just provide the number of the sector. Unless you are going to use the rest of the drive, as you have done in this case, specifying the `Last sector` is very rare. It is also fairly rare to specify `+sectors` (think of this as how many sectors you want to use for this partition), as sector sizes are somewhat confusing.

In most cases, you specify the size of a partition by using `+size` (think of this as how much space you want to allocate to the partition). A value of `+2000K`will create a 2000-kilobyte partition. A value of `+200M` will create a 200-megabyte partition. A value of `+2G` will create a 2-gigabyte partition.

At the `Command` prompt, type the `p` command to display the current partition table. You should see your new partition:`p` It is always a good idea to check your work before saving the changes. The `p` command shows the current partition table, but changes haven't been made to the hard drive yet.

Now you can create a partition that you can later format with a filesystem and access via a directory by mounting it. To create this new partition, first type `n` at the `Command` prompt:`n` At the next prompt, enter `+200M` to create a 200MB partition:`+200M` Don't forget to type the `+` character before `200M`.
Type `p` at the `Command` prompt to see your new partition:
Type `w` at the `Command` prompt to save your new partitions:
The kernel needs to be instructed to re-read the partition table that is now on the hard drive and put it into memory. You can tell that the kernel doesn't know about these new partitions yet because of the output of the following command:`ls /dev/sd*`
When the kernel recognizes new partitions, this will result in new device files being automatically created in the `/dev` directory. As you can see from the previous output, the kernel doesn't know about the two new partitions as there isn't a `/dev/sda2` or `/dev/sda5` file.Reboot the system to have the kernel recognize the new partitions:In some cases, you may be able to run a command, such as `kpartx` or `partprobe`, to have the kernel read the new partition table. However, on this system, these commands do not exist.
Then, verify the new files have been created in the `/dev` directory:`ls /dev/sd*`
Execute the following command to create an ext4 partition on the `/dev/sdb5`partition: `mkfs -t ext4 /dev/sdb5`
Create a directory to use as a mount point for the new partition by using the `mkdir` command:`mkdir /data`
Then, mount the new partition and verify that it mounted correctly by executing the `mount` commands as shown:`mount /dev/sdb5 /data mount`
Use the `nano` editor to edit the `/etc/fstab` file:`nano /etc/fstab`
`/dev/sdb5  /data  ext4  defaults  1  1`

The `/etc/fstab` file will be used to mount filesystems automatically when the system reboots. Verify this change with the following `tail` command:`tail -n 1 /etc/fstab`
If no output is displayed for the command above, verify that the `/etc/fstab` file does not contain additional empty lines and attempt the command again. Alternatively, you can increase the number of lines value for the `-n` option to the `tail` command.
When the system reboots, the `/etc/fstab` file will be used to mount filesystems automatically. However, you don't want to reboot the system since any error in this file could cause the boot process to fail completely. To test the new entry in this file, unmount the `/data` filesystem and remount it by only specifying the mount point:`umount /data  mount /data  mount`
When you only specify the mount point argument when executing the `mount` command, the command looks at the `/etc/fstab` file for the rest of the information (the partition to mount and the mount options).

If you made a mistake in the new entry in the `/etc/fstab` file, you might get an error like the following:

```
root@ubuntu:~# mount /data
mount: can't find /data in /etc/fstab or /etc/mtab
```

If this happens, look at the `/etc/fstab` file, correct the error, and try to mount again.

To create additional swap space, you either need to create a new partition or a new, large file. In the first example, you will create a swap partition. Start by using the `fdisk` command as shown:`fdisk /dev/sdb`
Create a new partition by entering n at the `Command` prompt and use the rest of the information provided below:`n`
Change the partition type by entering the `t` command at the `Command` prompt and entering the values provided below. When finished, enter the `p` command at the `Command` prompt to verify that your new partition has an `Id` of `82`:Type `w` at the Command prompt to save your new partition:`w`
In some cases, you may be able to run a command, such as the `partprobe` command, to have the kernel read the new partition table. However, on this system, this command does not exist.
hen, verify the new files have been created in the `/dev` directory:`ls /dev/sd*`
To format the partition as swap space with a label of `myswap`, execute the following `mkswap` command:`mkswap -L myswap /dev/sdb6`
Use the `free` command to see the current swap space for the system. Then, use the `swapon` command to add the new swap partition to current swap space. Finally, use the `free` command again to verify the changes: `free | swapon -a /dev/sdb6| free`
Use the `nano` editor to edit the `/etc/fstab` file:`nano /etc/fstab`
To have the new swap partition enabled automatically at boot, add the following line to the bottom of the `/etc/fstab` file: `LABEL=myswap none swap sw 0 0`
Verify this change with the following `tail` command:`tail -n 1 /etc/fstab`
To test the new setting in the `/etc/fstab` file, you could reboot the system. However, if you made any errors, the system may end up being unbootable. A better solution for testing the new entry in the `/etc/fstab` file is to execute the `swapon -a` after first removing the partition from swap space. Execute the following commands: `swapon -s | swapoff /dev/sdb6|swapon -s|swapon -a |swapon -s`
In the commands above, the `swapon` command used with the `-s` option demonstrates which swap spaces are currently being used, while the swapon command used with the `-a` option enables all swap devices that are listed in the `/etc/fstab` file.If there were any errors in the line that you just added to this file, the `swapon -a` command would have produced output error messages. For example:Notice that the value for `LABEL` was mistyped (it should only have one `y`, not two). If you get an error message when executing the `swapon -a` command, review the entry in the `/etc/fstab` file, correct it and try to execute the `swapon -a` command again.

To create a new swap file, first, create a large file with the `dd` command. To determine a good location for this new file, run the `df -h` command to see which partition has enough space to hold the swap file:`df -h`
in tkupthe `/var` directory (which is part of the `/` partition):`dd if=/dev/zero of=/var/swapfile bs=1M count=100`
The result should be a file that is approximately 100MB in size. Confirm this by executing the following command:`ls -lh /var/swapfile.`
Add the swap file to the current swap space and then confirm by executing the following commands:
`swapon /var/swapfile | swapon -s`

Maintaining Integrity

Execute the following command to list filesystem disk space usage details:`df`
Sizes are given in 1K (kilobyte) block sizes. Knowing how to view this information is useful because a filesystem that is full can cause problems, as users and system processes will be unable to write to the filesystem.
To view the output of the `df` command in more human-readable format, use the `-h` option:`df -h`
To view filesystem inode usage, use the `-i` option:`df -i`
Recall that each file needs an inode. Therefore, if a filesystem has 1,310,720 inodes then this also represents the maximum number of files that can reside on that filesystem.
To determine which directories are using the most disk space, use the `du` command. For example, execute the following command to determine how much space the `/usr` directory is using:`du /usr | more`
The output of the `du` command can be immense. To see just a summary of how much space a specific directory structure is using, use the `-s` option: `du -s /usr`
The output is given in block sizes (1 block = 1 kilobyte, in this case). To see a more human-readable value, use the `-h` option:`du -sh /usr`
The `du` command is useful because once you discover that a filesystem is close to becoming full, you need to determine where the largest chunks of files are. A common way of doing this is to see which of the directories under the root directory are using the most space. Execute the following command to see a demonstration: `du -sh /*`
Execute the following command to find the largest files in the `/etc` directory structure:`du /etc | sort -nr | head`
In addition to displaying file space by filesystem or directory structure, you also want to know how to display other information about filesystems. Execute the following command to display filesystem information:`dumpe2fs /dev/sda1 | head`
This output contains very useful filesystem information. For example, to display the number of `Free inodes` (every file needs an inode, so the number of `Free inodes` indicates how many more files you can place on this filesystem), execute the following command:`dumpe2fs /dev/sda1 | grep "Free inodes" | head -1` Once a filesystem has been created, the number of total inodes is set in stone.

To see the default mount options of the filesystem, execute the following command:`dumpe2fs /dev/sda1 | grep "Default mount options"`The default mount options are specified when the filesystem is initially created. Any additional mount options are specified in the `/etc/fstab` file.

Most of a filesystem's attributes can only be specified when the filesystem is created. For example, the number of inodes cannot be changed later. However, the reserved block space can be changed. Execute the following command to see the current value of the reserved block count:`dumpe2fs /dev/sda1 | grep "Reserved block count"`

The reserved block count is how much space on the filesystem is reserved for the root account or processes that run as the root account (typically, system processes).On some filesystems, such as `/home`, you may want to reduce this value since the root user doesn't use that filesystem often. On other filesystems, you may find the need to increase the reserved block count.

To change the reserved block count, you change the percentage of the filesystem that is reserved. This reserved amount can only be used by the root account and is set to 5% by default. Execute the following commands to change this value and confirm the change:`dumpe2fs /dev/sda1 | grep "Reserved block count"`

```
tune2fs -m 10 /dev/sda1

dumpe2fs /dev/sda1 | grep "Reserved block count"
```

Fixing Filesystems

In order to complete this lab, a new partition must be created. The steps to create a new partition will essentially be the same as what you performed in a previous lab. Execute the following command to start this process:`fdisk /dev/sdb`
At the `Command` prompt, type `p` to display the current partition table:`p`
At the `Command` prompt, type `n` to create a new partition:`n`
At the `Select` prompt, type `e` to specify that you are going to create an extended partition:`e`

At the `Partition` number prompt, type `1` to specify that you are going to use the first partition number:`1`

At the `Command` prompt, type `p` to display the current partition table. You should see your new partition:`p`

Now, you can create a partition that you can later format with a filesystem and access via a directory by mounting it. To create this new partition, first type `n` at the `Command` prompt:`n`

Type `w` at the `Command` prompt to save your new partitions:`w`

Then, verify the new files have been created in the `/dev` directory:`ls /dev/sd*`

Execute the following command to create an ext4 partition on the `/dev/sdb5` partition:`mkfs -t ext4 /dev/sdb5`

Create a directory to use as a mount point for the new partition by using the `mkdir` command. Then, mount the new partition and verify that it mounted correctly by executing the mount commands as shown.`mkdir /data mount /dev/sdb5 /data`

Attempt to run the `fsck` utility on the `/dev/sdb5` filesystem by executing the following command:`fsck /dev/sdb5`.The `fsck` utility will not run on a mounted filesystem. The purpose of

the `fsck` command is to fix filesystem problems on filesystems that can't be mounted. If the filesystem is mounted, there are no filesystem problems and no need to run the `fsck` command.

Unmount the `/dev/sda5` partition and then attempt to run the `fsck` utility on the /dev/sda5 filesystem by executing the following commands:`umount /dev/sdb5`

`fsck /dev/sdb5`.The `clean` value indicates that this filesystem is either new or has been correctly unmounted. Therefore, there is no need to run `fsck` because the filesystem is not broken.

One way of having the `fsck` utility run on a `clean` filesystem is to force the checking of the filesystem with the `-f` option:`fsck -f /dev/sdb5`

Another technique that you can use is to manually switch the filesystem state from `clean` to `not clean`. To do this, execute the following command:`debugfs -w -R "ssv state 0" /dev/sdb5`

A filesystem state of `not clean` just indicates that the filesystem was not properly unmounted. There may be problems with the filesystem, but then again, there may be no problems at all. If a filesystem is set to `not clean`, then the `fsck` utility will check the filesystem.The `debugfs` command can make changes directly to the filesystem. Use this utility with caution on filesystems that are on critical systems, as you could easily damage the filesystem with this utility.

After switching the filesystem state, you can run the `fsck` command:`fsck /dev/sdb5`

The superblock is where critical filesystem data is stored. If the primary superblock is corrupted, then you need to use a backup superblock to fix the primary superblock. To view the backup superblocks of a filesystem, execute the following command:`dumpe2fs /dev/sdb5 | grep superblock`

The following command is intended for testing purposes. You would never run this command on a production machine. Execute the following command to force the corruption of your primary superblock:Be very careful when typing this command. If you don't type it exactly as shown, it may end up deleting the entire filesystem. If that happens, the only solution is to reset the VM and restart the lab.`dd if=/dev/zero of=/dev/sdb5 bs=1024 count=1 seek=1`

To verify that the filesystem now has a problem, attempt to mount the filesystem by executing the following command `mount /dev/sdb5 /data`Normally, the `mount` command can determine the filesystem type by looking at values in the primary superblock. However, if that information can't be accessed, you will get the error message `mount: you must specify the filesystem type.`

Execute the following command to try to mount the filesystem by specifying the filesystem type:`mount -t ext4 /dev/sdb5 /data`

Notice the suggestion to view the contents of the `dmesg` command. Execute the following command:`dmesg | tail`The highlighted lines demonstrate that without the primary superblock, the `mount`command has no way of finding the ext4 filesystem on this partition.

You can use a backup superblock to fix the primary superblock. Unfortunately, if the primary superblock is corrupted, then you can't use the `dumpe2fs` command to view the location of backup superblocks: `dumpe2fs /dev/sdb5 | grep superblock`

However, there is a second way to determine a backup superblock that will work on a filesystem that has a corrupted primary superblock. Execute the following command:`mke2fs -n /dev/sdb5`Do not forget to include the `-n` option. Without the `-n` option, you would end up creating a new filesystem on that partition, resulting in complete data loss.

To fix the filesystem using the backup superblock, execute the following command. When prompted with `Fix<y>?` type the letter **Y**:`fsck -b 8193 /dev/sdb5`

Verify that the filesystem was fixed by mounting it with the first `mount` command and then verify that the filesystem mounted properly with the second `mount` command:`mount /dev/sdb5 /data`

The most common time for a filesystem error to occur is during the boot process. In the last phase of this lab, you will create a filesystem error and fix it during the boot process. Start by using the `nano` editor to edit the `/etc/fstab` file:`nano /etc/fstab`

```
/dev/sdb5    /data ext4   defaults     0      0
```

If the `/etc/fstab` file is currently empty, make the line above the first and only line of the file.

Verify this is correct by executing the following command:Note that the character after the `-n` option is a number `1`, not a lowercase letter L.

To test that this line is accurate, unmount the `/dev/sda5` filesystem and remount it again by specifying only the mount point:`umount /data`Recall that when you specify only the mount point, the mount command looks at the corresponding entry in the `/etc/fstab` file to determine the rest of the information needed to mount. If the `mount` command is successful, the entry in the `/etc/fstab` file is correct.

Next, unmount the filesystem and use the `dd` command to create an error in the filesystem `umount /data`

```
dd if=/dev/zero of=/dev/sdb5 bs=1024 count=1 seek=1
```

Because this filesystem is a non-critical filesystem (not needed to boot the system), you could press the letter `S`, and the mounting process for this filesystem would be skipped. You could then login and fix the problem by running the `fsck` command. However, if it was a critical filesystem, you wouldn't be able to do that. For this lab, consider this a critical filesystem.

At this prompt, type the following command to fix the filesystem:`fsck /dev/sdb5`Unlike the `fsck` utility that you run when logged in as root, the `fsck` utility that runs when in maintenance mode, as is the case here, knows to check backup superblocks automatically.

Lastly, after you have finished running the `fsck` command, you should look in the `lost+found` directory for the filesystem to see if any lost files have been placed in that directory. Execute the following command:`ls /data/lost+found`

No output is a good thing. If there are any lost files, they will not have regular names; rather, they will be named after their inode number.

Package Management

Please use the CentOS image to complete the following steps
Display some of the installed software packages on the system by using the `head` command to display the first 10 packages. Execute the following command:`rpm -qa | head`
To see details about a specific RPM, execute the following command:`rpm -qi setup`Note that while the full name of the rpm is `setup-2.8.71-10.el7.src.rpm`, the version number and architecture can be omitted when referring to the package.
To view the scripts that are included with the package, execute the following command:`rpm -qi --scripts setup`
Execute the following command to view the documentation that was included in the setup package:`rpm -q -d setup`
Display the status of the package files by executing the following command:`rpm -q -s setup | head`
Remove the `quota` package from the system with the following `rpm` command:`rpm -e quota`
Recall that the `rpm` command doesn't check for any package dependencies. As a result, you would not see any warning if another package depended on the `quota` package.
Determine what the `quota` package requires in order for it to work correctly by executing the following command:`rpm -qp --requires /mnt/local_repo/quota* | head`
Install the `quota` package by executing the following command:`rpm -i /mnt/local_repo/quota*`The `quota` package is being installed from a local repository, a collection of packages that have been downloaded from the internet ahead of time for this lab.
To demonstrate the use of the rpm2cpio command, first remove the /usr/sbin/edquota file:`rm /usr/sbin/edquota` Remember that the `rpm2cpio` command is useful in that it will allow you to

extract files from the rpm without installing the files. This is useful to recover a single file from the package (such as when you accidentally delete a key software file like `edquota`).
Try to execute the `edquota` command:`edquota`
In most cases, you won't know what package a file belongs to. You can determine this if you know the full path to the package, but this also is something that you may not automatically know. If the file was installed as part of a package, you could execute the following command to determine the full path to the file:`rpm -q -a -s | grep edquota`
Based on the output of the previous command, you can now use the `rpm` command to determine which package provided the `/usr/sbin/edquota`file. Execute the following command:`rpm -q -f /usr/sbin/edquota`

Execute the following commands to extract all of the files from the `quota`package into the `/tmp` directory:`cd /tmp rpm2cpio /mnt/local_repo/quota* | cpio -imud`It isn't really necessary to extract all of the files, but there are a couple of reasons why you might. To begin with, using the `-imud` option to the `cpio` command is easier than trying to specify a single file to extract. Another reason to extract all of the files is that it is likely that more than one file is missing.
Execute the following `ls` command to view the new `edquota` file:`ls /tmp/usr/sbin`
Execute the following command to copy this file to the correct location:`cp /tmp/usr/sbin/edquota /usr/sbin`
Execute the following command to verify that the `edquota` command has been recovered:`edquota -v`Note that after recovering the file, you can delete all of the new files that were created. However, since you placed them in the `/tmp` directory, they would eventually be deleted since `old files` in the `/tmp` directory are routinely deleted.
Your virtual system has been configured to act as a YUM repository. In order to be able to use this repository, some changes need to be made to your system. Begin by moving all of the files from the `/etc/yum.repos.d` directory into the `/tmp` directory:`ls /etc/yum.repos.d`

```
mv /etc/yum.repos.d/* /tmp
ls /etc/yum.repos.d
ls /tmp
```

Normally, you want these files in the `/etc/yum.repos.d` directory as they allow you to connect to repositories on the internet or on local media devices (like your DVD drive). However, for this lab, these files conflict with the local repository.
Using the `vi` editor, create a file called `Local.repo` in the`vi /etc/yum.repos.d/Local.repo`
ype the letter `i` to enter Insert mode and then type the following:`name=Local`

```
baseurl=file:///mnt/local_repo
gpgcheck=0
enabled=1
```

The `baseurl` setting defines where the RPM files are stored. Normally this would be a network-based address, either http or ftp based. The `gpgcheck` setting is set to `0` to prevent the `yum` command from checking the digital signature of the packages. The `enabled` setting is set to `1` in order to make use of this repository.
Verify your work by executing the following command:`cat /etc/yum.repos.d/Local.repo`
Remove the `quota` package by executing the following `yum` command. When prompted `Is this ok`, press the **Y** key:`yum remove quota`

```
y
```

Verify that the `quota` package is available on the local repository by executing the following `yum` command:`yum list available`
Install the `quota` package by executing the following `yum` command. When prompted `Is this ok`, type the **Y** key:`yum install quota`
Ubuntu image to complete the following steps.
Debian's package management system is based upon the format used to package the software files for the Debian distribution; these package file names end in `.deb`. The Debian package management system is used by many other distributions including Ubuntu and Mint Linux. These `.deb` files are archives that contain the software and the information to configure it for installation or removal.
The Debian package management system can install software from repositories listed in `/etc/apt/sources.list`. In a networked environment, these Internet sites can usually be reached. In this virtual environment, none of the repositories are reachable. Use the `tail` command to view a portion of this file:`tail /etc/apt/sources.list`
Since none of the repositories are reachable, rename the `/etc/apt/sources.list` file so a new file that points to a local repository can be created during the next step:`mv /etc/apt/sources.list /etc/apt/sources.list.orig`
Execute the following `echo` command to create a single repository in a new `/etc/apt/sources.list` file. This repository exists on the local filesystem in the `/var/www/debs/amd64` directory.`echo 'deb [arch=amd64 trusted=yes] file:/var/www/debs amd64/' > /etc/apt/sources.list`
Now that the `/etc/apt/sources.list` contains a reachable repository, execute the the following command to receive the list of packages that are available:`apt-get update`
To install the updated versions of all available packages, execute the command below:`apt-get upgrade` Based on the previous output, zero packages were upgraded, newly installed, removed, or not upgraded. The output also recommends removing the `htop` package as it is no longer needed.
Remove the `htop` package using the following command. When prompted whether to continue, press **Enter** to continue:`apt-get remove htop`
To find packages to install, you can use `apt-cache search keyword`. Search for packages related to `apt` by executing the following command:`apt-cache search apt`
View the dependencies of the `aptitude` package by executing the following command:`apt-cache depends apt`
View the details of the `aptitude` package by executing the following command:`apt-cache show apt`
Unlike removing a package, purging will also remove configuration files used by the package. Purge the `xfsprogs` package by executing the following command. When prompted whether to continue, press the **Enter** key to continue:`apt-get purge xfsprogs`
Installing a package can be accomplished by executing the `apt-get install` command followed by the package name. Execute the following command to install the `xfsprogs` package. When prompted whether to continue, press the **Y** key and then press the **Enter** key to continue:`apt-get install xfsprogs.`

While using Advanced Packaging Tools (APT) commands offers features like automatic dependency resolution and remote repositories, the `dpkg` command can also be used to remove or install software packages. Use the `dpkg` command to remove the `xfsprogs` package:`dpkg -r xfsprogs`

The `dpkg` command can also be used to install software packages. Use the `dpkg` command to install the `xfsprogs` package:`dpkg -i /var/www/debs/amd64/xfsprogs_4.9.0+nmu1ubuntu2_amd64.deb`

The `dpkg` command is also useful for listing all of the packages that are currently installed on the system by executing the `dpkg -l` command. An additional argument can be provided as a glob pattern; this will only list packages that match that pattern. List all the packages that match the glob `apt*`: `dpkg -l 'apt*'`

The status of an installed package can be displayed by executing the `dpkg -s` command followed by the name of the package. Display the status of the `xfsprogs` package:`dpkg -s xfsprogs`
SUSE image to complete the following steps
The SUSE package management system is based on the ZYpp/libzypp package management engine, which is mainly implemented by openSUSE, SUSE Linux Enterprise and Ark. ZYpp/Libzypp is the background package management engine for the `zypper` command line tool. This package management utility provides users with the ability to query, install, remove, and update software packages as well as the ability to satisfy software dependencies during these actions. In this portion of the lab, you will practice using the `zypper` command to manage packages on the OpenSUSE operating system.
he syntax used for the `zypper` command examples in this course is the following `zypper [--global-opts] command [--command-opts] [command-arguments]`
o begin using the `zypper` command, first ensure that the command has updated information about repositories by using the following command:`zypper ref`
To query the software repositories on a system, use the list repositories `-lr` option can be used with the `zypper` command. Use the following command to list the repositories:`zypper lr`

To find a package to install, the search `se` command can be used to query the configured repositories on the system. Use the following command to search for the `python` package:`zypper se python`
Use the following command to search for the `cowsay` package:`zypper se cow*`
The `zypper` command can be used with the install `in` command to install packages.`zypper in package_name`
Run the following command to view package status, it should show `i` for installed:`zypper se cowsay`
Like other Linux programs, the `cowsay` command has many advanced features:

Managing Shared Libraries

Shared libraries are important as they contain code that will be used by multiple executable programs. An issue related to accessing one shared library may have a serious impact on multiple system commands and processes. As a result, while not a common task, it is important to know how to manage these libraries.

Execute the following commands to verify that there are no customizations to the `ldconfig` command on this system:`more /etc/ld.so.conf`

`ls /etc/ld.so.conf.d`

Execute the following command to display all of the libraries that the `ldconfig` utilizes:`ldconfig -v`

To add a new library directory to the system, first create a new directory and then create a configuration file in the `/etc/ld.so.conf.d` directory:

`mkdir /usr/mylib`

`echo "/usr/mylib" > /etc/ld.so.conf.d/mylib.conf`

Normally, the next step would be to copy the library files into the `/usr/mylib` directory. In this case, we will copy an existing one into this directory:`cp /lib64/ld-2.17.so /usr/mylib`

To verify that this library has been added to the system, execute the following command:`ldconfig -v | head -5`

Delete the previously created `mylib.conf` configuration file in the `/etc/ld.so.conf.d` directory:`rm /etc/ld.so.conf.d/mylib.conf`

Now, verify that the directory is no longer used by the `ldconfig` command:`ldconfig -v | grep mylib`

In most cases, if you are deleting the configuration file permanently, then you could also delete the directory that contains the libraries (`/usr/mylib` in this case).

Display the libraries that are used by the `/bin/ls` command by executing the following command:`ldd /bin/ls`