

What Is The Shell?

When we speak of the command line, we are really referring to the shell. The shell is a program that takes keyboard commands and passes them to the operating system to carry out. Almost all Linux distributions supply a shell program from the GNU Project called bash.

The name “bash” is an acronym for “Bourne Again SHell”, a reference to the fact bash is an enhanced replacement for sh, the original Unix shell program written by Steve Bourne.

Command History

If we press the up-arrow key, we will see that the previous command reappears after the prompt. This is called command history. Most Linux distributions remember the last 1000 commands by default. Press the down-arrow key and the previous command disappears.

Command list

Command	Description
Date	<code>\$date</code> Sat Dec 28 12:55:50 IST 2019 This command displays the current time and date. <code>date +%H</code> Shows Hour e.g 11 <code>date +%m</code> Shows month number e.g. 12 <code>date +%M</code> Shows minute e.g. 23 <code>date +%y</code> Shows year (2 digit)e.g. 19 <code>date +%Y</code> Shows year (4 digit)e.g. 2019 <code>date +%Z</code> Shows time zone e.g IST <code>date +%T</code> Shows current time in HH:MM:SS format 16:48:23 <code>date +%D</code> Shows current date in DD/MM/YY format 28/12/19
cal	Displays calendar for current month of current year <code>\$cal</code> December 2019 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 <code>\$cal 2018</code> Display calendar for the year 2018.
echo	<code>echo</code> command in linux is used to display line of text/string that are passed as an argument. <code>-e</code> here enables the interpretation of backslash escapes <code>-n</code> : this option is used to omit echoing trailing newline

	<pre>\$echo "welcome \nto\n linux" welcome \nto\n linux \$echo -e "welcome \nto\n linux" welcome to linux \$ echo "hi";echo "welcome" hi welcome \$ echo -n "hi";echo -n "welcome" hiwelcome</pre>
bc	<p>bc command is used for command line calculator</p> <p>The bc command supports the following features:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Arithmetic operators <input type="checkbox"/> Increment or Decrement operators <input type="checkbox"/> Assignment operators <input type="checkbox"/> Comparison or Relational operators <input type="checkbox"/> Logical or Boolean operators <input type="checkbox"/> Math functions <input type="checkbox"/> Conditional statements <input type="checkbox"/> Iterative statements <pre>Input : \$ echo "12+5" bc Output : 17 Input : \$ echo "10^2" bc Output : 100</pre>
exit	We can end a terminal session by either closing the terminal emulator window, or by entering the exit command at the shell prompt
pwd	To display the current working directory, we use the pwd (print working directory) command.
ls	To list the files and directories in the current working directory
mkdir	<p>Create Directories</p> <p>mkdir dir1 would create a single directory named "dir1", while mkdir dir1 dir2 dir3 would create three directories named "dir1", "dir2", and "dir3".</p>
cp	<p>Copy Files And Directories</p> <p>cp file1 file2 Copy file1 to file2. If file2 exists, it is overwritten with the contents of file1. If file2 does not exist, it is created.</p>

	<p>cp -i file1 file2 Same as above, except that if file2 exists, the user is prompted before it is overwritten.</p> <hr/> <p>cp file1 file2 dir1 Copy file1 and file2 into directory dir1. dir1 must already exist.</p> <hr/> <p>cp dir1/* dir2 Using a wildcard, all the files in dir1 are copied into dir2. dir2 must already exist.</p>
mv	<p>Move And Rename Files The mv command performs both file moving and file renaming.</p> <p>mv file1 file2 Move file1 to file2. If file2 exists, it is overwritten with the contents of file1. If file2 does not exist, it is created. In either case, file1 ceases to exist.</p> <hr/> <p>mv -i file1 file2 Same as above, except that if file2 exists, the user is prompted before it is overwritten.</p> <hr/> <p>mv file1 file2 dir1 Move file1 and file2 into directory dir1. dir1 must already exist.</p> <hr/> <p>mv dir1 dir2 If directory dir2 does not exist, create directory dir2 and move the contents of directory dir1 into dir2 and delete directory dir1. If directory dir2 does exist, move directory dir1 (and its contents) into directory dir2.</p>
rm	<p>Remove Files And Directories</p> <p>rm file1 Delete file1 silently.</p> <p>rm -i file1 Same as above, except that the user is prompted for confirmation before the deletion is performed.</p> <p>rm -r file1 dir1 Delete file1 and dir1 and its contents.</p> <p>rm -rf file1 dir1 Same as above, except that if either file1 or dir1 do not exist, rm will continue silently</p>
	<p><u>Be Careful With rm!</u> Unix-like operating systems such as Linux do not have an undelete command. Once you delete something with rm, it's gone. Linux assumes you're smart and you know what you're doing. IF YOU ARE A NEW USER TO LINUX COMMAND LINE, YOU SHOULD ALWAYS USE rm -i. You can specify multiple files</p>
rm -i filename	Asks you before every file removal for confirmation.
rm -R dir-name	Will remove the directory dir-name recursively.
rm -rf dir-name	Will remove the directory dir recursively, ignoring non-existent files and will never prompt for anything. BE CAREFUL USING THIS COMMAND!
rmdir dir-name	Will remove the directory dir-name, if it's empty. This command can only remove empty directories.

mkdir dir-name	Create a directory dir-name.
touch filename	Create a file filename, if it doesn't exist, otherwise change the timestamp of the file to current time.
cd	cd .. Go to the parent directory of current directory (mind the space between cd and ..) cd . Current directory cd / Take you to root directory of the system cd Take you to HOME DIRECTORY
rmdir dir-name	Will remove the directory dir-name, if it's empty. This command can only remove empty directories
touch filename	Create a file filename, if it doesn't exist, otherwise change the timestamp of the file to current time
man ls	Read the manual page of ls command
Help	In Bash shell, this will display the list of all available bash commands.
whatis date	List a one-line description of date command
hostname	Display hostname of the system
passwd	Change password of current user
whoami	Username of the users logged in at the terminal
who	List of all the users currently logged in as a user
chmod	Changing permissions - read,write,execute of a file or directory
top	List all processes sorted by their current system resource usage. Displays a continually updated display of processes (By default 3 seconds). Use q key to exit top
ps	List processes currently running on current shell session Ps -aux List all the processes by all users on the current system ps -u user List all of the processes and commands user is running
find /var/www -name '*.css'	This will print out the full path/filename to all files under /var/www that end in .css.
grep font /var/www/html/style.css	This will print all lines containing the pattern font in the specified file. grep -R font /var/www/html/ grep -R will recursively access the directory
gzip	gzip command compresses files. Each single file is compressed into a single file f given a file as an argument, gzip compresses the file, adds a ".gz" suffix, and deletes the original file \$ gzip -v mydoc.txt

	<p>OUTPUT :</p> <p>new.txt: 18.2% -- replaced with new.txt.gz</p>
Gunzip	<p>gunzip command is used to compress or expand a file or a list of files in Linux. It accepts all the files having extension as .gz, .z, _z, -gz, -z, .Z, .taz or.tgz and replace the compressed file with the original file by default. The files after uncompression retain its actual extension.</p> <p>Input: \$gunzip filename.txt.gz</p> <p>Output: filename.txt</p>
Tar	<p>The Linux 'tar' stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important command which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.</p> <p>Options:</p> <ul style="list-style-type: none"> -c : Creates Archive -x : Extract the archive -f : creates archive with given filename -t : displays or lists files in archived file -u : archives and adds to an existing archive file -v : Displays Verbose Information -A : Concatenates the archive files -z : zip, tells tar command that create tar file using gzip -j : filter archive tar file using tbzip -W : Verify a archive file -r : update or add file or directory in already existed .tar file <hr/> <p>1. Creating an uncompressed tar Archive using option -cvf : This command creates a tar file called file.tar which is the Archive of all .c files in current directory.</p> <p>\$ tar cvf file.tar *.c</p> <p>Output :</p> <p>os2.c os3.c os4.c</p> <hr/> <p>2. Extracting files from Archive using option -xvf : This command extracts files from Archives.</p> <p>\$ tar xvf file.tar</p> <p>Output :</p> <p>os2.c os3.c os4.c</p> <hr/> <p>3. gzip compression on the tar Archive, using option -z : This command creates a tar file called file.tar.gz which is the Archive of</p>

	.c files. \$ tar cvzf file.tar.gz *.c <hr/> 4. Extracting a gzip tar Archive *.tar.gz using option -xvzf : This command extracts files from tar archived file.tar.gz files. \$ tar xvzf file.tar.gz
Finger	The finger displays information about the system users. \$finger dotcom Output : Login: dotcom Name: dotcom Directory: /home/dotcom Shell: /bin/bash On since Sat Dec 28 10:16 (IST) on :0 from :0 (messages off) No mail. No Plan.
Mesg	Control write access to you terminal \$mesg y : will enable your terminal to receive the message \$mesg n : will Prevents the display of terminal messages from other users. \$mesg : will display current status [y / n]
Mail	The mail command invokes the standard sendmail binary (/usr/sbin/sendmail) which in turns connects to the local MTA to send the mail to its destination. The local MTA is a locally running smtp server that accepts mails on port 25. \$ mail -s "Hello World" someone@example.com Cc: Hi Peter How are you I am fine Good Bye <Ctrl+D> OR \$ mail -s "This is the subject" somebody@example.com <<< 'This is the message'
Talk	The "talk" command allows you to talk to other users on the same system
Write	write command in Linux is used to send a message to another user.
Wall	wall command in Linux system is used to write a message to all users
Who who command without any arguments, it will display account information (user login	\$ who user1 tty1 2019-12-16 19:27 root tty2 2019-12-16 19:26 remote1 pts/1 2019-12-16 19:27 (192.168.56.1)

name, user's terminal, time of login as well as the host the user is logged in from)	
who -h To print the heading of the columns displayed,	<pre>\$ who -H NAME LINE TIME COMMENT user1 tty1 2019-12-16 19:27 user2 pts/0 2019-12-16 19:26 (192.168.56.1) root pts/1 2019-12-16 19:27 (192.168.56.1)</pre>
who -q To print the login names and total number of logged on users.	<pre>\$ who -q User1 remote1 root # users=3</pre>
who -b To view the time of last system boot.	<pre>\$ who -b system boot 2019-12-16 02:39</pre>
who -a It will print combined information as we discussed above	<pre>\$ who -a system boot 2019-12-16 02:39 run-level 3 2018-01-19 02:39 LOGIN tty1 2018-01-19 02:39 3258 id=1 LOGIN ttyS0 2018-01-19 02:39 3259 id=S0</pre>
ps Shows the processes for the current shell -	<pre>[root@dotcom ~]# ps PID TTY TIME CMD 12330 pts/0 00:00:00 bash 21621 pts/0 00:00:00 ps Where PID - the unique process ID TTY - terminal type that the user is logged into TIME - amount of CPU in minutes and seconds that the process has been running CMD - name of the command that launched the process.</pre>
ps -r	view all running processes [root@dotcom ~]# ps -r
ps -x	view all processes owned by you [root@dotcom ~]# ps -x
ps -f	view full format listing [root@dotcom ~]# ps -af tux 17327 17326 0 12:42 pts/0 00:00:00 -bash

	tux 17918 17327 0 12:50 pts/0 00:00:00 ps -af
wc	<p>wc stands for word count. As the name implies, it is mainly used for counting purpose.</p> <p>It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.</p> <p>By default it displays four-columnar output.</p> <p>First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument</p>
wc -l	<p>-l: This option prints the number of lines present in a file.</p> <p>With one file name</p> <pre>\$ wc -l state.txt 5 state.txt</pre> <p>With more than one file name</p> <pre>\$ wc -l state.txt capital.txt 5 state.txt 5 capital.txt 10 total</pre> <p>-w This option prints the number of words present in a file</p> <p>-c This option displays count of bytes present in a file.</p>
cat	<p>To view single file \$cat filename1</p> <p>To view multiple files \$cat file1, file2, file3</p> <p>To create a file \$cat > filename</p> <p>Type some text ad Pres ctrl +d to save and exit</p>
uname	<p>Print system information</p> <p>uname -a Print all information like hostname, machine type, os release, version etc.</p>
ln	<p>- Create Links</p> <p>The ln command is used to create either hard or symbolic links. It is used in one of two ways:</p> <p>ln file link to create a hard link, and:</p> <p>ln -s item link to create a symbolic link where "item" is either a file or a directory.</p> <p>Hard Links</p> <p>Hard links are the original Unix way of creating links, compared to symbolic links, which are more modern. By default, every file has a single hard link that gives the file its name. When we create a hard link, we create an additional directory entry for a file. Hard links have two important limitations:</p> <ol style="list-style-type: none"> 1. A hard link cannot reference a file outside its own file system. This means a link cannot reference a file that is not on the same disk partition as the link itself. 2. A hard link may not reference a directory.

	<p>Symbolic Links</p> <p>Symbolic links were created to overcome the limitations of hard links. Symbolic links work by creating a special type of file that contains a text pointer to the referenced file or directory. In this regard, they operate in much the same way as a Windows shortcut though of course, they predate the Windows feature by many years ;-)</p> <p>A file pointed to by a symbolic link, and the symbolic link itself are largely indistinguishable from one another. For example, if you write something to the symbolic link, the referenced file is written to. However when you delete a symbolic link, only the link is deleted, not the file itself. If the file is deleted before the symbolic link, the link will continue to exist, but will point to nothing. In this case, the link is said to be broken. In many implementations, the ls command will display broken links in a distinguishing color, such as red, to reveal their presence.</p> <p>Symbolic links were created to overcome the two disadvantages of hard links: Hard links cannot span physical devices and hard links cannot reference directories, only files. Symbolic links are a special type of file that contains a text pointer to the target file or directory.</p>
--	--

Filters Pipelines are often used to perform complex operations on data. It is possible to put several commands together into a pipeline. Frequently, the commands used this way are referred to as filters.

- cat - Concatenate files
- sort - Sort lines of text
- uniq - Report or omit repeated lines
- grep - Print lines matching a pattern
- wc - Print newline, word, and byte counts for each file
- head - Output the first part of a file [by default first 10 lines of a file]
head -15 filename will display first 15 lines of a file
- tail - Output the last part of a file [by default last 10 lines of a file]
tail -15 filename will display last 15 lines of a file
- tee - Read from standard input and write to standard output and files

Important Facts About Filenames

1. Filenames that begin with a period character are hidden. This only means that ls will not list them unless you say ls -a. When your account was created, several hidden files were placed in your home directory to configure things for your account. Later on we will take a closer look at some of these files to see how you can customize your environment. In addition, some

applications place their configuration and settings files in your home directory as hidden files.

2. Filenames and commands in Linux, like Unix, are case sensitive. The filenames "File1" and "file1" refer to different files.
3. Linux has no concept of a "file extension" like some other operating systems. You may name files any way you like. The contents and/or purpose of a file is determined by other means. Although Unix-like operating systems don't use file extensions to determine the contents/purpose of files, many application programs do.
4. Though Linux supports long filenames which may contain embedded spaces and punctuation characters, limit the punctuation characters in the names of files you create to period, dash, and underscore. Most importantly, do not embed spaces in filenames. If you want to represent spaces between words in a filename, use underscore characters. You will thank yourself later.

Absolute Pathnames: An absolute pathname begins with the root directory and follows the tree branch by branch until the path to the desired directory or file is completed.

Relative Pathnames: Where an absolute pathname starts from the root directory and leads to its destination, a relative pathname starts from the working directory. To do this, it uses a couple of special notations to represent relative positions in the file system tree. These special notations are "." (dot) and ".." (dot dot).

The "." notation refers to the working directory and the ".." notation refers to the working directory's parent directory.

Pipelines

The ability of commands to read data from standard input and send to standard output is utilized by a shell feature called pipelines. Using the pipe operator "|" (vertical bar), the standard output of one command can be piped into the standard input of another:

```
command1 | command2
$ ls -l /usr/bin | less
```

ls command and its options

Option	Long Option	Description
-a	-all	List all files, even those with names that begin with a period, which are normally not listed (i.e., hidden).
-A	--almost	-all Like the -a option above except it does not list . (current directory) and .. (parent directory).
-d	--directory	Ordinarily, if a directory is specified, ls will list the contents of

		the directory, not the directory itself. Use this option in conjunction with the -l option to see details about the directory rather than its contents.
-F	--classify	This option will append an indicator character to the end of each listed name. For example, a "/" if the name is a directory.
-h	--human-readable	In long format listings, display file sizes in human readable format rather than in bytes.
-l		Display results in long format.
-r	--reverse	Display the results in reverse order. Normally, ls displays its results in ascending alphabetical order.
-S		Sort results by file size.
-t		Sort by modification time.

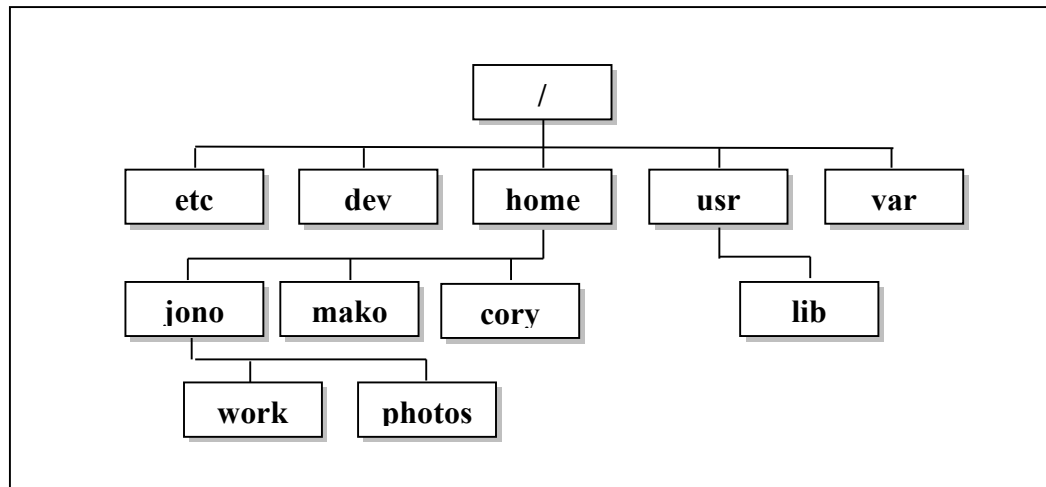
[directory list in long format and sorted on time]

```
$ls -lt
total 12
-rw-r--r-- 1 root UsersGrp 354 Dec 23 13:01 file1.sh
drwxr-xr-x 1 root UsersGrp 10 Dec 20 17:24 book
-rw-r--r-- 1 root UsersGrp 25 Dec 20 17:14 abc
-rw-r--r-- 1 root UsersGrp 20 Nov 28 12:59 demo1
```

As we saw before, the "-l" option causes ls to display its results in long format. This format contains a great deal of useful information. Here is the Examples directory from an early Ubuntu system:

Field	Meaning
-	The first character indicates the type of file. Among the different types, a leading dash means a regular file, while a "d" indicates a directory.
rw-r--r--	Access rights to the file. The next three characters are the access rights for the file's owner, the next three are for members of the file's group, and the final three are for everyone else.
1	File's number of hard links. See the discussion of links later in this chapter.
Root	The username of the file's owner.
UsersGrp	The name of the group which owns the file.
354	Size of the file in bytes.
Dec 23 13:01	Date and time of the file's last modification.
file1.sh	File name

Linux file system: linux file system name is ext2 [eXtended file system], the directory structure is inverted tree. Where root is at the top
Basic explanation of structure hierarchy:



Directory	Comments
/root	This is the home directory for the root account.
/proc	The /proc directory is special. It's not a real file system in the sense of files stored on your hard drive. Rather, it is a virtual file system maintained by the Linux kernel. The "files" it contains are peepholes into the kernel itself. The files are readable and will give you a picture of how the kernel sees your computer.
/dev	Linux exposes devices as files, and the /dev directory contains a number of special files that represent devices . These are not actual files as we know them, but they appear as files – for example, /dev/sda represents the first SATA drive in the system
/tmp	The /tmp directory is intended for storage of temporary, transient files created by various programs. Some configurations cause this directory to be emptied each time the system is rebooted.
/etc	ETC is a folder which contain all your system configuration files in it. Then why the etc name? "etc" is an English word which means etcetera i.e in layman words it is "and so on"
/bin	The /bin directory contains the essential user binaries (programs) that must be present when the system is mounted
/sbin	This directory contains "system" binaries. These are programs that perform vital system tasks that are generally reserved for the superuser.
/usr OR /home	The /usr directory tree is likely the largest one on a Linux system. It contains all the programs and support files used by regular users.
/usr/bin	/usr/bin contains the executable programs installed by your Linux distribution. It is not uncommon for this directory to hold thousands of programs.
/usr/lib	The shared libraries for the programs in /usr/bin.
/usr/local	The /usr/local tree is where programs that are not included with your

	distribution but are intended for systemwide use are installed. Programs compiled from source code are normally installed in /usr/local/bin. On a newly installed Linux system, this tree exists, but it will be empty until the system administrator puts something in it.
/usr/sbin	Contains more system administration programs.
/usr/share	/usr/share contains all the shared data used by programs in /usr/bin. This includes things like default configuration files, icons, screen backgrounds, sound files, etc.
/usr/share/doc	Most packages installed on the system will include some kind of documentation. In /usr/share/doc, we will find documentation files organized by package. 22 A Guided Tour Directory Comments
/var	With the exception of /tmp and /home, the directories we have looked at so far remain relatively static, that is, their contents don't change. The /var directory tree is where data that is likely to change is stored. Various databases, spool files, user mail, etc. are located here.
/var/log	/var/log contains log files, records of various system activity. These are very important and should be monitored from time to time. The most useful ones are /var/log/messages and/or /var/log/syslog. Note that for security reasons on some systems, you must be the superuser to view log files

Command	Utility

Q-1 ???????????????

Q-2 ???????????

- 3 **Write a shell script to execute following commands**
1. **Sort file abc.txt and save this sorted file in xyz.txt**
 2. **Give an example of : To execute commands together without affecting result of each other.**
 3. **How to print "this is a three -line
 1. Text message"**
 4. **Which command display version of the UNIX?**
 5. **How would u get online help of cat command?**

Ans. echo "sorting the file"
 sort abc.txt > xyz.txt
 echo "executing two commands"
 who ; ls
 echo -e "this is \n a three-line \n Text message"
 echo "The version is `uname -a`"
 echo "Help of cat command"
 man cat

- 4 **Write a shell script to execute following commands**
1. **How would u display the hidden files?**
 2. **How delete directory with files?**
 3. **How would user can do interactive copying?**
 4. **How would user can do interactive deletion of files?**
 5. **Explain two functionality of "mv" command with example?**

Ans. echo "1. How would u display the hidden files"
 echo "2. How delete directory with files"
 echo "3. How would user can do interactive copying"
 echo "4. How would user can do interactive deletion of files"
 echo "5. Explain two functionality of "mv" command with example"
 echo "enter your choice"
 read ch
 case \$ch in
 1) echo "Displaying hidden files"
 ls .[a-z]* ;;
 2) echo "Deleting directories with files"
 rm -R dirname
 3) echo "Interactive copy"
 cp -i file1 file2 ;; # file2 should be created first to check interactivity
 4) echo "Interactive Deletion"
 rm -i file1 ;;
 5) echo "mv command"
 mv oldfilename newfilename ;;
 *) echo "Invalid choice" ;;

 esac

- 5 **Write a shell script to execute following commands**
1. **Create a file called text and store name,age and address in it.**
 2. **Display the contents of the file text on the screen.**
 3. **Delete the directories mydir and newdir at one shot.**
 4. **Sort a numeric file?**
 5. **Change the permissions for the file newtext to 666.**

Ans. echo "1. Create a file called text and store name,age and address in it."
 echo "2. Display the contents of the file text on the screen."
 echo "3. Delete the directories mydir and newdir at one shot."
 echo "4. Sort a numeric file"
 echo "5. Change the permissions for the file newtext to 666."
 echo "enter your choice"
 read ch
 case \$ch in
1) echo "Create a file called text and store name,age and address in it."
 echo "Enter the filename"
 read fn
 cat > \$fn ;;
2) echo "Display the contents of the file text on the screen."
 cat \$fn ;;
3) echo "Delete the directories mydir and newdir at one shot."
 rmdir mydir newdir ;;
4) echo "Sort a numeric file"
 sort -n filename ;;
5) echo "Change the permissions for the file newtext to 666."
 chmod 666 newtext ;;
*) echo "Invalid choice" ;;
 esac

- 6 **Write shell script that accept filename and displays last modification time if file exists, otherwise display appropriate message.**

Ans. if [-e \$fn]
 then
 ls -l \$fn | cut -d " " -f8 #change the column number for desired output
 else
 echo "File does not exist"
 fi

- 7 **Write a shell script to display the login names that begin with 's'.**

Ans. who | grep ^s

- 8 **Write a shell script to remove the zero sized file from the current directory**

Ans.

```
for i in *
do
if [ ! -s $i ]
then

rm $i
echo " $i removed "
fi
done
```

OR

```
$find . -size 0 -delete
```

9 Write a shell script to display the name of all the executable file from the current directory.

Ans.

```
for i in *
do
if [ -x $i ]
then
countx=`expr $countx + 1`
echo $i
fi
echo "Number of executable files are $countx"
```

10 Write a shell script that will display welcome message according to time

Ans.

```
d=`date +%H`
if [ $d -lt 12 ]
then
echo "Good Morning"
elif [ $d -gt 12 -a $d -lt 14 ]
then
echo "Good Afternoon"
else
echo "Good Evening"
fi
```

11 Write a shell script to find number of ordinary files and directory files.

Ans.

```
for i in *
do
if [ -d $i ]
then
countd=`expr $countd + 1`
```



```

fi
if [ -f $i ]
then
    countf=`expr $countf + 1`
fi
done
echo "Number of directories are $countd "
echo "Number of Ordinary files are $countf"

```

- 12 Write a shell script that takes a filename from the command line and checks whether the file is an ordinary file or not.**
- If it is an ordinary file then it should display the contents of the file.
 - If it is not an ordinary file then script should display the message:
"File does not exist or is not ordinary, cannot display."

Ans.

```

if [ -f $1 ]
then
    echo "It is an ordinary file"
    cat $1
else
    echo "File does not exist or is not ordinary file"
fi

```

- 13 Write a shell script that takes a filename from the user and checks whether it is a directory file or not.**
- If it is a directory, then the script should display the contents of the directory.
 - If it is not a directory file then script should display the message:
"File is not a directory file"

Ans.

```

echo "enter the filename"
read fn

if [ -d $fn ]
then
    echo "Its a directory"
    ls $fn
else
    echo "Its not a directory"
fi

```

- 14 Write a shell script that takes a filename as an argument and checks if the file exists and is executable.**
- If the file is executable then the shell script should display the message: "File exists"

- If the file does not exist and is not executable then the script should display the message: "File does not exist or is not executable."

Ans.

```
echo "enter the filename"
read fn

if [ -e $fn -a -x $fn ]
then
    echo "file exists and is executable"
else
    echo "file does not exist or is not executable"
fi
```

- 15 Write a shell script that displays all subdirectories in current working directory.**

Ans.

```
echo "List of Directories. "
for i in *
do
    if [ -d $i ]
    then
        echo $i
    fi
```

OR

```
find . -type d -print
```

- 16 Write a shell script that calculates the number of ordinary and directory files in your current working directory.**

Ans.

```
for i in *
do
    if [ -d $i ]
    then
        countd=`expr $countd + 1`
    fi
    if [ -f $i ]
    then
        countf=`expr $countf + 1`
    fi
done
echo "Number of directories are $countd "
echo "Number of Ordinary files are $countf"
```

- 17 Write a shell script that accepts 2 filenames and checks if both exist; if both exist then append the content of the second file into the first file.**

Ans.

```
echo "enter the first filename"
read fn1
echo "enter the second filename"
read fn2
if [ -f $fn1 -a -f $fn2 ]
then
    echo "Both file exists"
    cat $fn2 >> $fn1
else
    echo "Files does not exist"
fi
```

18 Write a shell script that takes the name of two files as arguments and performs the following:

i. Displays the message :

"Displaying the contents of file :(first argument)"and displays the contents page wise.

ii. Copies the contents of the first argument to second argument.

iii. Finally displays the message :

"File copied successfully."

Ans.

```
echo "Displaying the contents of file $1"
cat $1
echo "Displaying the contents page wise"
cat $1 | more
echo "Copying the files"
cp $1 $2
c=`echo $?`
if [ $c -eq 0 ]
then
    echo "File copied successfully"
else
    echo "Files not copied successfully"
fi
```

19 Write a shell script to display the following menu and acts accordingly:

i. Calendar of the current month and year.

ii. Display "Good Morning/Good Afternoon/Good Evening" according to the current login time.

iii. User name, Users home directory.

iv. **Terminal name, Terminal type.**

v. **Machine name.**

vi. **No. of users who are currently logged in; List of users who are currently logged in.**

Ans.

```
echo "Menu"
echo "1. Calendar of the current month and year."
echo "2. Display "Good Morning/Good Afternoon/Good Evening" according to the
current login time."
echo "3. User name, Users home directory."
echo "4. Terminal name, Terminal type."
echo "5 Machine name."
echo "6. No. of users who are currently logged in; List of users who are currently
logged in."
echo "enter your choice"
read ch
case $ch in
1) echo "Calendar of current month is"
   cal ;;
2) d=`date +"%H"`
   if [ $d -lt 12 ]
   then
       echo "Good Morning"
   elif [ $d -gt 12 -a $d -lt 16 ]
   then
       echo "Good Afternoon"
   else
       echo "Good Evening"
   fi
3) echo "Username is $USER"
   echo "Users Home directory is $HOME" ;;
4) echo "Terminal details"
   tty;;
5) echo "Machine name is"
   uname -m ;;
6) echo "The number of users logged in are"
   who | wc -l
*) echo "Invalid choice"
esac
```

20 **Write a shell script that displays the following menu and acts accordingly**

1. **Concatenates two strings**
2. **Renames a file**
3. **Deletes a file.**
4. **Copy the file to specific location**

Ans.

```
echo "1. Concatenates two strings "
echo "2. Renames a file"
echo "3. Deletes a file."
echo "4. Copy the file to specific location"
echo "enter your choice"
read ch
case $ch in
1) echo "enter first string"
   read str1
   echo "enter second string"
   read str2
   echo "The concated strings are $str1$str2" ;;
2) echo "enter the old filename"
   read ofn
   echo "enter the new filename"
   read nfn
   mv $ofn $nfn
   echo "file renamed" ;;
3) echo "enter the filename"
   read fn
   rm $fn
   echo "file deleted" ;;
4) echo "enter the filename"
   read fn
   cp $fn \usr\home\dir\ $fn  #you can change the specific path
   echo "file copied" ;;
*) echo "invalid choice" ;;
esac
```

21 Write a shell script to change the suffix of all your *.txt files to .dat.

Ans.

```
for file in *.txt
do
mv $file `echo $file | sed 's/ \(.*\.\ \) txt/\1dat/'` ;
done
```

OR

```
for file in $1/*.dat ; do mv "$file" "${file%.*}.txt" ; done
```

22 Write a shell script to accept a directory-name and display its contents. If input is not given then HOME directory's contents should be listed. (Make use of command line argument)

Ans.

```

if [ $# ]
then
    ls $1
else
    ls $HOME
fi

```

To execute do as below

```
$sh filename.sh dir1
```

- 23 Write a shell script to get all files of home directory and rename them if their names start with c. e.g Newname = oldname111**

Ans.

```

Suffix="111"
shift
for f in "$@"
do
    extension=${f##*.}
    if [ -z $extension ]; then
        mv "$f" "$f$Suffix"
    else
        mv "$f" "${f%.$extension}$Suffix.$extension"
    fi
done

```

- 24 Write a shell script that takes two filename as arguments. It should check whether the contents of two files are same or not, if they are same then second file should be deleted.**

Ans.

```

echo "enter the first filename"
read fn1
echo "enter the second filename"
read fn2
cmp $fn1 $fn2
c=`echo $?`
if [ $c -eq 0 ]
then
    echo "both files are same"
    rm $fn2
else
    echo "both files are not same"
fi

```

```
$sh fn.sh dir1 dir2
```

- 25 Write a shell script that accepts two directory names from the command line and copies all the files of one directory to another.
The script should do the following
- If the source directory does not exist, flash an error message
 - If destination directory does not exist create it
 - Once both exist copy all the files from source directory to destination directory.

Ans.

```
if [ $# ]
then
    if [ -d $1 ]
    then
        if [ -d $2 ]
        then
            cp -R $1 $2
        else
            mkdir $2
            echo "Directory created $2"
            cp -R $1 $2
        fi
    else
        echo "source directory does not exist"
    fi
else
    echo "Please provide command line arguments"
fi
```

- 26 Write a shell script that displays the following menu
- List home directory
 - Date
 - Print working directory
 - Users logged in

Read the proper choice. Execute corresponding command. Check for invalid choice.

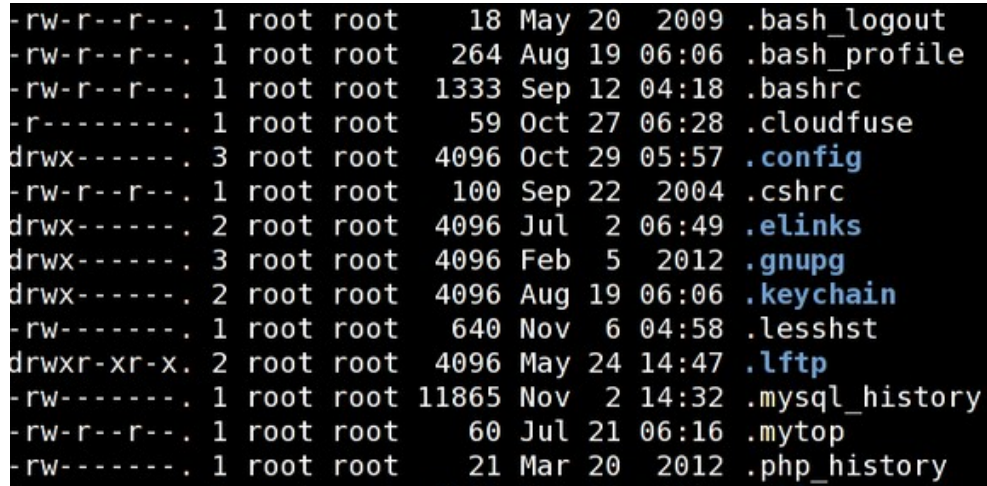
Ans.

```
echo "1.List home directory"
echo "2.Date"
echo "3. Print working directory"
echo "4. Users logged in"
echo "enter your choice"
read ch
case $ch in
1) echo "Home directory is $HOME" ;;
2) echo "Todays date is `date` " ;;
3) echo "Present working directory is `pwd` " ;;
4) echo " No of users logged in are"
    who ;;
```

```
*) echo "Invalid choice" ;;  
esac
```

27 Write a shell script that displays all hidden files in current directory.

Ans. `$ ls -a | egrep '^\.'`



A screenshot of a terminal window showing the output of the command `ls -a | egrep '^\.'`. The output lists hidden files and their permissions, owner, group, size, and modification date. The files listed are: `.bash_logout`, `.bash_profile`, `.bashrc`, `.cloudfuse`, `.config`, `.cshrc`, `.elinks`, `.gnupg`, `.keychain`, `.lessht`, `.lftp`, `.mysql_history`, `.mytop`, and `.php_history`. The file `.config` is highlighted in blue.

OR

`ls .[a-z]*`

28 Write a shell script that Combine two files in the third file horizontally and vertically.

Ans.

```
echo "enter the first filename"  
read fn1  
echo "enter the second filename"  
read fn2  
  
echo "Combining two files horizontally"  
cat $fn2 >> $fn1  
  
echo "Combining two files vertically"  
paste $fn1 $fn2
```

29 Write a shell script to delete all the spaces from a given file.

Ans.

```
echo "enter the filename"  
read datafile  
cat $datafile | tr -d '[:space:]' > newfile
```


30 Write a shell script to find a given date fall on a weekday or a weekend.

Ans.

```
d=`date +%u`  
if [ $d -eq 7 ]  
then  
    echo "it is weekend"  
else  
    echo "it is a weekday"  
fi
```

31 Write a shell script to search for a given word in all the files given as the arguments on the command line.

Ans.

```
echo "Enter the word"  
read w  
for i in $@  
do  
    grep $w $i  
done
```

32 Write a shell script that display last modified file in the current directory.

Ans. *ls -lt | head -2 | tail -1*

33 Write a script to display the permissions of the particular file.

Ans.

```
echo "enter the filename"  
read fn  
  
if [ -r $fn ]  
then  
    echo "Read permission"  
fi  
if [ -x $fn ]  
then  
    echo "eXecute permission"  
fi  
if [ -w $fn ]  
then  
    echo "Write permission"  
fi
```

OR

```
ls -l $fn | cut -c 2-10
```

- 34 Write a shell script to display the calendar in the following manner:
- Display the calendar of months m1 and m2 by 'CAL m1, m2' command file.
 - Display the calendar of the months from m1 to m2 by 'CAL m1-m2' command file.

Ans.

```
m1=$1
y1=$2
wat=$3
m2=$4
y2=$5
if test $wat = ","
then
    cal $m1 $y1;cal $m2 $y2

elif test $wat = "-"
then
    while [ $y1 -le $y2 ]
    do

        cal $m1 $y1
        read a
        if [ $y1 -eq $y2 -a $m1 -ge $m2 ]
        then
            exit
        fi
        m1=`expr $m1 + 1`
        if [ $m1 -gt 12 ]
        then
            m1=1
            y1=`expr $y1 + 1`
        fi
    done
else
    echo Syntax Mth1 Yr1 [,-/] Mth2 Yr2
fi

# run the shell script as
sh -x calrange.sh 1 2019 , 11 2019
sh -x calrange.sh 1 2019 - 11 2019
sh -x calrange.sh 1 2019 # 11 2019
```

- 35 Write a shell script to display the following menu for a particular file :
- Display all the words of a file in ascending order.
 - Display a file in descending order.
 - Toggle all the characters in the file.
 - Display type of the file.

Ans. `echo "1. Display all the words of a file in ascending order."
 echo "2. Display a file in descending order."
 echo "3. Toggle all the characters in the file."
 echo "4. Display type of the file."
 echo "enter your choice"
 read ch
 echo "enter the filename"
 read fn
 case $ch in
 1) sort $fn;;
 2) sort -r $fn;;
 3) cat $fn | tr "[a-z][A-Z]" "[A-Z][a-z]"
 4) file $fn;;
 *) echo "invalid choice"
 esac`

36 Write a shell script to check whether the named user is currently logged in or not.

Ans. `echo "enter the username"
 read un
 c=`who | grep -c $un`
 if [$c -gt 0]
 then
 echo "User is currently logged in "
 else
 echo "User is not currently logged in"
 fi`

37 Write a shell script to display the following menu for a particular file:

- i. Display all the words of a file in ascending order.**
- ii. Display a file in descending order.**
- iii. Display a file in reverse order.**
- iv. Toggle all the characters in the file**
- v. Display type of the file.**

Ans. `echo "1.Display all the words of a file in ascending order."
 echo "2.Display a file in descending order."
 echo "3.Display a file in reverse order."
 echo "4.Toggle all the characters in the file"
 echo "5.Display type of the file."
 echo "Enter your choice"
 read ch
 echo "enter the file name"
 read fn
 case $ch in
 1) sort $fn ;;`

```

2) sort -r $fn ;;
3) rev $fn
4) cat $fn | tr "[a-z][A-Z]" "[A-Z][a-z]"
5) file $fn
*) echo "Invalid choice"
esac

```

38 Write a shell script to find total no. Of users and finds out how many of them are currently logged in.

Ans. `echo "The number of users in the system are"`
`cat /etc/passwd | wc -l`
`echo "The number of uses currently logged in are "`
`who | wc -l`
`or`

```

total=`cat /etc/passwd | wc -l`
cur_log=`who | wc -l`
echo "Total users      : $total"
echo "Currently logged : $cur_log"

```

**39 Write a shell script that displays the directory information in the following format-
Filename Size Date Protection Owner**

Ans. `x=`ls -l | wc -l``

`i=2`

`echo "Filename\t\t\tSize\tDate\tProtection\tOwner\n"`

`while [$i -le $x]`

`do`

`s=`ls -l | head -$i | tail -1 | tr -s " "``

`fn=`echo $s | cut -d " " -f 9``

`si=`echo $s | cut -d " " -f 5``

`d1=`echo $s | cut -d " " -f 6``

`d2=`echo $s | cut -d " " -f 7``

`p=`echo $s | cut -d " " -f 1``

```
o=`echo $s | cut -d " " -f 3`

echo "$fn\t\t\t$si\t$d1 $d2\t$p\t$o"

i=`expr $i + 1`

done
```

OR

```
echo "Enter the filename"
read fn

echo " Filename   Size   Date   Protection   Owner"
echo "`ls -l $fn | cut -d ' ' -f3` `ls -l $fn | cut -d ' ' -f5` `ls -l $fn | cut -d ' ' -f6,7` `ls -l $fn | cut -d ' ' -f1` `ls -l $fn | cut -d ' ' -f4` "
```

40 Write a shell script to display five largest files from the current directory

Ans. `ls -lS | head -6 | tail -1`

41 Write a shell script that toggles contents of the file

Ans. `echo "Enter the filename"`
`read fn`
`cat $fn | tr "[a-z][A-Z]" "[A-Z][a-z]"`

42 Write a shell script that report whether your friend has currently logged in or not.

Ans. `echo "Enter the username"`
`read un`

```
c=`who | grep -c $un`

if [ $c -gt 0 ]
then
    echo "User is currently logged in "
else
    echo "User is not currently logged in"
fi
```

44 Write a shell script to accept any character using command line and list all the files starting with that character in the current directory

Ans. `ls | grep ^$1`

run the shell script as: sh filename.sh a
it will list files starting with a

- 45 Create a file called student containing roll-no, name and marks.**
- a. Display the contents of the file sorted by marks in descending order**
 - b. Display the names of students in alphabetical order ignoring the case.**
 - c. Display students according to their roll nos.**
 - d. Sort file according to the second field and save it to file 'names'.**
 - e. Display the list of students who scored between 70 and 80.**

Ans. echo "enter the filename"
read fn
cat > \$fn

enter roll-no | name | marks of students and press ctrl+d

echo "1. Display the contents of the file sorted by marks in descending order"
echo "2. Display the names of students in alphabetical order ignoring the case."
echo "3. Display students according to their roll nos."
echo "4. Sort file according to the second field and save it to file 'names'."
echo "5. Display the list of students who scored between 70 and 80"

echo "enter your choice"
read ch

case \$ch in
1) sort -k5 -r \$fn ;;
2) sort -k3 -i \$fn ;;
3) sort \$fn ;;
4) sort -k3 \$fn > names ;;
5) awk '{ if(\$5 > 70 && \$5 < 80) print \$5 }' \$fn ;;
*) echo "Invalid Choice"
esac

File Management Commands

Linux uses some conventions for present and parent directories. This can be a little confusing for beginners. Whenever you are in a terminal in Linux, you will be in what is called the current working directory. Often your command prompt will display either the full working directory, or just the last part of that directory. Your prompt could look like one of the following:

```
user@dotcom ~/somedir $  
user@ dotcom somedir $  
user@ dotcom /home/user/somedir $  
which says that your current working directory is /home/user/somedir.
```

In Linux .. represents the parent directory and . represents the current directory.

Therefore, if the current directory is /home/user/somedir, then `cd ../somedir` will not change the working directory.

The table below lists some of the most used file management commands

File/directory permissions and groups

Command Utility

GoalKicker.com – Linux® Notes for Professionals 5

`chmod <specification> filename` Change the file permissions. Specifications = u user, g group, o other, + add permission, - remove, r read, w write, x execute.

`chmod -R <specification> dirname`

Change the permissions of a directory recursively. To change permission of a directory and everything within that directory, use this command.

`chmod go+=r myfile` Add read permission for the owner and the group.

`chmod a +rwx myfile` Allow all users to read, write or execute myfile.

`chmod go -r myfile` Remove read permission from the group and others.

`chown owner1 filename` Change ownership of a file to user owner1.

`chgrp grp_owner filename` Change primary group ownership of file filename to group grp_owner.

`chgrp -R grp_owner dir-name`

Change primary group ownership of directory dir-name to group grp_owner recursively. To change group ownership of a directory and everything within that directory, use this command.

Section 1.3: Hello World

Type the following code into your terminal, then press Enter :

```
echo "Hello World"
```

This will produce the following output:

```
Hello World
```

Section 1.4: Basic Linux Utilities

Linux has a command for almost any tasks and most of them are intuitive and easily interpreted.

Getting Help in Linux

Command Usability

`man <name>` Read the manual page of <name>.

`man <section> <name>` Read the manual page of <name>, related to the given section.

`man -k <editor>` Output all the software whose man pages contain <editor> keyword.

`man -K <keyword>` Outputs all man pages containing <keyword> within them.

`apropos <editor>` Output all the applications whose one line description matches the word ditor.

When not able to recall the name of the application, use this command.

`help` In Bash shell, this will display the list of all available bash commands.

`help <name>` In Bash shell, this will display the info about the <name> bash command.

`info <name>` View all the information about <name>.

`dpkg -l` Output a list of all installed packages on a Debian-based system.

`dpkg -L packageName` Will list out the files installed and path details for a given package on Debian.

`dpkg -l | grep -i <edit>` Return all .deb installed packages with <edit> irrespective of cases.

`less /var/lib/dpkg/available` Return descriptions of all available packages.

`whatis vim` List a one-line description of vim.

<command-name> --help Display usage information about the <tool-name>. Sometimes command -h also works, but not for all commands.

User identification and who is who in Linux world Command Usability hostname Display hostname of the system.

GoalKicker.com – Linux® Notes for Professionals 6
hostname -f Displays Fully Qualified Domain Name (FQDN) of the system.

passwd Change password of current user.

whoami Username of the users logged in at the terminal.

who List of all the users currently logged in as a user.

w Display current system status, time, duration, list of users currently logged in on system and other user information.

last Who recently used the system.

last root When was the last time root logged in as user.

lastb Shows all bad login attempts into the system.

chmod Changing permissions - read,write,execute of a file or directory.

Process related information

Command Usability

top List all processes sorted by their current system resource usage. Displays a continually updated display of processes (By default 3 seconds). Use q key to exit top.

ps List processes currently running on current shell session

ps -u root List all of the processes and commands root is running

ps aux List all the processes by all users on the current system

Section 1.5: Searching for files by patterns in name/contents

A common task of someone using the Linux Command Line (shell) is to search for files/directories with a certain name or containing certain text. There are 2 commands you should familiarise yourself with in order to accomplish this:

Find files by name

find /var/www -name '*.css'

This will print out the full path/filename to all files under /var/www that end in .css. Example output:

/var/www/html/text-cursor.css

/var/www/html/style.css

For more info:
man find
Find files containing text
grep font /var/www/html/style.css

This will print all lines containing the pattern font in the specified file. Example output:

```
font-weight: bold;
```

```
font-family: monospace;
```

Another example:

```
grep font /var/www/html/
```

GoalKicker.com – Linux® Notes for Professionals 7

This doesn't work as you'd hoped. You get:

```
grep: /var/www/html/: Is a directory
```

You need to grep recursively to make it work, using the -R option:

```
grep -R font /var/www/html/
```

Hey nice! Check out the output of this one:

```
/var/www/html/admin/index.php: echo '<font color=red><b>Error: no  
dice</b></font><br/>';
```

```
/var/www/html/admin/index.php: echo '<font color=red><b>Error: try  
again</b></font><br/>';
```

```
/var/www/html/style.css: font-weight: bold;
```

```
/var/www/html/style.css: font-family: monospace;
```

Notice that when grep is matching multiple files, it prefixes the matched lines with the filenames.

You can use the -
h option to get rid of that, if you want.

For more info:

man grep

Section 1.6: File Manipulation

Files and directories (another name for folders) are at the heart of Linux, so being able to create, view, move, and delete them from the command line is very important and quite powerful. These file manipulation commands allow you to perform the same tasks that a graphical file explorer would perform.

Create an empty text file called myFile:

```
touch myFile
```

Rename myFile to myFirstFile:

```
mv myFile myFirstFile
```

View the contents of a file:

```
cat myFirstFile
```

View the content of a file with pager (one screenful at a time):

```
less myFirstFile
```

View the first several lines of a file:

head myFirstFile

View the last several lines of a file:

tail myFirstFile

Edit a file:

GoalKicker.com – Linux® Notes for Professionals 8

vi myFirstFile

See what files are in your current working directory:

ls

Create an empty directory called myFirstDirectory:

mkdir myFirstDirectory

Create multi path directory: (creates two directories, src and myFirstDirectory)

mkdir -p src/myFirstDirectory

Move the file into the directory:

mv myFirstFile myFirstDirectory/

You can also rename the file:

user@linux-computer:~\$ mv myFirstFile secondFileName

Change the current working directory to myFirstDirectory:

cd myFirstDirectory

Delete a file:

rm myFirstFile

Move into the parent directory (which is represented as ..):

cd ..

Delete an empty directory:

rmdir myFirstDirectory

Delete a non-empty directory (i.e. contains files and/or other directories):

rm -rf myFirstDirectory

Make note that when deleting directories, that you delete ./ not / that will wipe your whole filesystem.

Section 1.7: File/Directory details

The ls command has several options that can be used together to show more information.

Details/Rights

The l option shows the file permissions, size, and last modified date. So if the root directory contained a dir called test and a file someFile the command:

GoalKicker.com – Linux® Notes for Professionals 9

user@linux-computer:~\$ ls -l

Would output something like

-rw-r--r-- 1 user users 70 Jul 22 13:36 someFile.txt

drwxrwxrwx 2 user users 4096 Jul 21 07:18 test

The permissions are in format of drwxrwxrwx. The first character represents the file type d if it's a directory -otherwise. The next three rwx are the permissions the user has over the file, the next three are the permissions the group has over the file, and the last three are the permissions everyone else has over the file.

The r of rwx stands for if a file can be read, the w represents if the file can be modified, and the x stands for if the file can be executed. If any permission isn't granted a - will be in place of r, w, or x.

So from above user can read and modify someFile.txt but the group has only read-only rights.

To change rights you can use the `chmod ### fileName` command if you have sudo rights. r is represented by a value of 4, w is represented by 2, and x is represented by a 1. So if only you want to be able to modify the contents to the test directory

Owner rwx = $4+2+1 = 7$

Group r-x = $4+0+1 = 5$

Other r-x = $4+0+1 = 5$

So the whole command is

`chmod 755 test`

Now doing a `ls -l` would show something like

`drwxr-xr-x 2 user users 4096 Jul 21 07:20 test`

Readable Size

Used in conjunction with the l option the h option shows file sizes that are human readable.

Running

`user@linux-computer:~$ ls -lh`

Would output:

`total 4166`

`-rw-r--r-- 1 user users 70 Jul 22 13:36 someFile.txt`

`drwxrwxrwx 2 user users 4.0K Jul 21 07:18 test`

Hidden

To view hidden files use the a option. For example

`user@linux-computer:~$ ls -a`

Might list

GoalKicker.com - Linux® Notes for Professionals 10

`.profile`

`someFile.txt`

`test`

Total Directory Size

To view the size of the current directory use the s option (the h option can also be used to make the size more

readable).

`user@linux-computer:~$ ls -s`

Outputs

`total 4166`

`someFile.txt test`

Recursive View