Objective:- Assignments will help trainees to understand the basics of unix command and how to explore and use it.

NOTE:- Always use "man" command before using any unix commands and read about it and its options and how to use it.

Assignments:-

1. List the contents of a directory and their attributes

   Answer: ls command list the files and directories and ls -lh command show detailed information.

2. Long list the content with file, directory ownership, permissions,sizes, etc…

   Answer: ls -l command

3. Display the size of the file in human readable format

   Answer: ls -lh

4. Show all files and folders including hidden one

   Answer: ls -a

5. list directories recursively

   Answer: ls -R

6. Sort the files by size with largest at the top

   Answer: ls -laSh

7. Sort the files by last time modified displaying the newest first.

   Answer: ls -lt

8. Diplay the location of a program/command, where it is installed.

   Answer: whereis

9. Which command is used to switch directory from one to another

Answer: cd

10. List all the environment variables set for the current shell environment

Answer: Use the command printenv or env

11. Did you notice something in the output of "env" command?

Answer: It prints only the environment variables.

12. Which command is used to print the text or any variables value in the Console/Terminal?

Answer: echo

13. Print the value of the env variable "PATH" on the console

Answer: echo “$PATH”

14. Is linux a case-sensitive operating system?

Answer: Yes

15. Is, "Ls" same as "ls"?

Answer: No it is case sensitive language

16. Display your currently logged in user

Answer: w or who

17. how do you change the currrently logged in user to another user?

Answer: su

18. Which command is used to leave a shell environment that you are currently logged in to?

Answer: exit

19. How do you reboot the system?

Answer: First we should sign as root account by sudo su then type sudo reboot

20. How do you shutdown the system?

Answer: sudo shutdown -n now

21. Display all the major running processes in the system

Answer: ps

22. Understand the output of command used in above question of displaying processes, explain the meaning of each column and what data it displays?

Answer: ps command displays four column first is PID which tells the process identification number,second is TTY type of terminal process is running on, third is TIME denotes total amount of cpu usage, and CMD name of command that started the process.

23. Display the name of the system kernel

Answer: uname -o

24. display the kernel release number

Answer: uname -r

25. display teh machine type of the current kernel

Answer: **uname -m**

26. Display the name of the operating system that the kernel is running on

Answer: uname -o

27. Display all info that uname command can show.

Answer: uname -a

28. Display the name of directory that you are currently pointing to

Answer: pwd

29. change the current directory to another directory that you have in your system.

Answer: cd directory_name

30. Go up one directory

Answer: cd .//

31. Return to last directory

Answer: cd ..

32. change the current directory to home(logged in user's) directory

Answer: cd /home

33. How to check all the command used from the prompt (Command History)

Answer: history

34. In which file the history of commands are stored in?

Answer: .bash_history

35. How many lines of history does the system keep and from where you can change it?

Answer: By default 1000 lines of codes are stored in history and stored in $HISTSIZE and $HISTFILESIZE variables.

36. How can you modify bash's history behaviour

Answer: By making changes to ~/.bashrc file.

37. Display all the commands entered so far, now, try to run a particular command from the history list without typing that command.

Answer: open the file using cat ~/.bash_history to display all the commands and to run a particular command from history use the ! followed by the entry number

NOTE:- TAB key is your friend when it comes to command completion and having long file and directory names autocompleted at the bash prompt for you. JUST BE LAZY AND USE TAB KEY FOR AUTO COMPLETION ;-)

38. What are the different types of shell and where are they used and how do we use them?

Answer: Shell is a program which provides the interface between the user and an operating system. Different types of shell are Bourne Shell, C shell , Korn Shell and Bourne again shell.

The Bourne shell served two primary goals:

- **Executing unix/linux commands for the operating system,i.e, *command line interpreter***
- **Writing reusable scripts that could be invoked through the shell,i.e, *scripting***

C shell

objective of achieving a scripting language similar to C programming language. This was useful given that C was a primary language in use back then which also made it easier and faster to use.

Korn shell

It is backward-compatible with the Bourne Shell. It included features from the C Shell such as job control, command aliasing & command history.

Bourne again shell(bash)

It showed all features from the Bourne shell but is much more efficient and easy to use.

It supported filename globbing, piping, command substitution, and control structures for conditional testing and iteration.

39. What is the difference between login shell and non-login shell?

Answer: A Login shell is created after a successful login of user. For example, when you login to a Linux system via terminal, SSH or switch to user with "su -" command.

Non Login Shell is the shell, which is started by the login shell. For example, A shell which you started from another shell or started by a program etc.

40. When a login shell start, it runs a set of predefined script to configure shell environment.

Answer: A Login Shell executes following scripts:

- **Login shell executes /etc/profile**
- **/etc/profile executes all scripts in /etc/profile.d/**
- **Then executes users ~/.bash_profile**
- **~/.bash_profile executes users ~/.bashrc**
- **~/.bashrc executes /etc/bashrc**

41. How do we start login shell and non-login shell?

Answer: A Login shell is created after a successful login of user and Non Login Shell is the shell, which is started by the login shell. For example, A shell which you started from another shell or started by a program.

42. What happens when you start a login shell (which files are read and used and Why)?

Answer: When Bash is invoked as an interactive login shell, or as a non-interactive shell with the --login option, it first reads and executes commands from the file /etc/profile , if that file exists. After reading that file, it looks for ~/. bash_profile , ~/. bash_login , and ~/.

43. What happens when you start a non-login shell (Which files are read and used and Why)?

Answer: When an interactive shell that is not a login shell is started, Bash reads and executes commands from ~/. bashrc , if that file exists. This may be inhibited by using the --norc option. The --rcfile file option will force Bash to read and execute commands from file instead of ~/.

44. What are Shell Configuration Files, why do we need it?

Answer: Shell configuration files are executed automatically when you log in and. out of a shell. They initialize and configure a shell upon login. Configuration files ("config files" for short) are important to modern computing. They allow you to customize how you interact with an application or how an application interacts with the rest of your system.

45. Explain the Order of file usage from the system/user's home directory when user logs in to the System.

Answer: /home/user

46. What are Shell Variables, list major shell variables and what do they represent?

Answer: A shell variable is a variable that is available only to the current shell.

Environment variables – Variables that are exported to all processes spawned by the shell. Their settings can be seen with the env command.
Shell (local) variables – Variables that affect only the current shell.

47. How we see all our env variables?

Answer: Use the command env or printenv

48. How we see all env variables in alphabical order?

Answer: use get-childitem env:* | sort-object Key

49. What Format does the env var and its values are stored?

Answer: The Global environment variables of your system are stored in /etc/environment.

User level Environment variables are mostly stored in .bashrc and .profile files in your Home folder.

50. How do you create your own varible?

Answer: variable_name=value

51. How do you start a new bash shell?

Answer: ctrl+alt+t

52. Difference between Local/Shell variables to Global Variable

Answer: **The difference between the two is that variables values may change during execution, while constant values cannot be reassigned. An environment variable is a variable whose value is set outside the program, typically through functionality built into the operating system or microservice.**

53. Making a variable accessible from other the shell in the system.

Answer: If you want to make a variable available to every new process, then you can put it in /etc/environment.

54. Show the real life use case of exporting a variable

Answer: environment variables are set when you open a new shell session. at any time if you change any of the variable values, the shell has no way of picking that change. The export command, on the other hand, provides the ability to update the current shell session about the change you made to the exported variable.

55. Convert the above script file into a command, The file should run with "myappp" instead of "myapp.sh"

Answer: Add the Shebang at very top i.e #!/bin/bash, use chmod u+x to make script executable then place the script into user/local/bin folder and finally run the script by just using scriptname.

56. What is Globbing? Explain in depth with examples?

Answer: Bash does not support native regular expressions like some other standard programming languages. The Bash shell feature that is used for matching or expanding specific types of patterns is called globbing. Globbing is mainly used to match filenames or searching for content in a file. Globbing uses wildcard characters to create the pattern. The most common wildcard characters that are used for creating globbing patterns are described below

'?' is used to match any single character. You can use '?' for multiple times for matching multiple characters.

Suppose, you want to search those text filenames whose names are 4 characters long and extension is .txt. You can apply globbing pattern by using '?' four times to do this task. Find out the list of all files and folder of the current directory using ls -l and then run the command ????.txt whose file name are four characters long and unknown.

57. List all entries with extension ".sh"

Answer: file . -name "*.sh".

58. List all entries with numbers in it.

Answer: grep -Eo '[0-9]{1,4}' filename.extension

59. List all entries that starts with a character and ends with a number

Answer: grep '^[a-z].*[0-9]$' filename.extension

60. List all entries that name length more than 5 characters

Answer: ls -A -d ?????*

61. What is Quoting? and Why do we need it?

Answer: In Linux Shell, many special characters have their own special meanings. Sometimes they are used to perform an action while other times they are just used as a character, so the quoting mechanism performs this task it makes us use them in whatever way we want to.

62. Write few(minimum 3) unique examples that shows, how a particular problem is solved using Quoting.

Answer: Consider an echo command that contains many special shell characters –

echo <-$1500.**>; (update?) [y|n]

Putting a backslash in front of each special character is tedious and makes the line difficult to read −

```
echo \<-\$1500.\*\*\>\; \(update\?\) \[y\|n\]
```

There is an easy way to quote a large group of characters. Put a single quote (') at the beginning and at the end of the string −

```
echo '<-$1500.**>; (update?) [y|n]'
```

Characters within single quotes are quoted just as if a backslash is in front of each character. With this, the echo command displays in a proper way.

If a single quote appears within a string to be output, you should not put the whole string within single quotes instead you should precede that using a backslash (\) as follows −

```
echo 'It\'s Shell Programming.
```

Example 2

```
VAR=ZARA

echo '$VAR owes <-$1500.**>; [ as of (`date +%m/%d`) ]'
```

Upon execution, you will receive the following result −

```
$VAR owes <-$1500.**>; [ as of (`date +%m/%d`) ]
```

This is not what had to be displayed. It is obvious that single quotes prevent variable substitution. If you want to substitute variable values and to make inverted commas work as expected, then you would need to put your commands in double quotes as follows −

```
VAR=ZARA

echo "$VAR owes <-\$1500.**>; [ as of (`date +%m/%d`) ]"
```

Upon execution, you will receive the following result −

```
ZARA owes <-$1500.**>; [ as of (07/02) ]
```

Example 3

The date command is executed in the following example and the

produced result is stored in DATA variable.

```
DATE=`date`

echo "Current Date: $DATE"
```

Upon execution, you will receive the following result −

```
Current Date: Thu Jul  2 05:28:45 MST 2009
```

63. How do you find a particular files/directories based on a particular search criteria?

    Answer: find . - name thisfile.txt. If you need to know how to find a file in Linux called thisfile. ...

find /home -name *.jpg. Look for all . jpg files in the /home and directories below it.

find . - type f -empty. Look for an empty file inside the current directory.

find /home -user randomperson-mtime 6 -iname ".d

HINT:- look for commands -> locate, find and whereis

64. Write major difference between locate, find and whereis?

Answer: find is what you use when you want to search by particular criteria and also manipulate files. It has many more options than locate so allows far more fine-grained control of results. It is slow because it performs the requested test(s) on every file to see if it matches.

locate is used to scan the whole system quickly for something - you might use this when you have no idea where something is, or when you want to find all related files scattered across various unknown places. It's fast because it uses a binary database to index the system. To get new files to show up, first run sudo updatedb (the database it updated once per day by cron

the whereis command simply returns the location of the executables, the man pages and the sources of a program.

65. How Globbing is different from locate, find and whereis?

Answer: Globbing is mainly used to match filenames or searching for content in a file. Globbing uses wildcard characters to create the pattern. The most common wildcard characters that are used for creating globbing patterns are described below

'?' is used to match any single character. You can use '?' for multiple times for matching multiple characters.

66. Explain the Linux File System.

Answer: Linux file system follows a tree-like hierarchical structure starting at the root. It consists of directories, sub-directories, and data files. This structure follows a standard layout recommended by Filesystem Hierarchy Standard (FHS), which is a standard maintained by the Linux Foundation.

Unlike Windows which has multiple roots, the Linux only allows one root.The root directory is where all other directories and files on the system reside and is designated by a forward slash /.

/home directory

home directory, a.k.a login directory, is where every user stores their personal files and documents.

/bin and /sbin directories

bin is short for binary. This is where Linux keeps its basic programs and applications. Binary files are the executable files that contain compiled source code. Almost all basic Linux commands can be found here such ls, cat, touch, pwd, rm, echo, … The binaries on this directory must be available in order to attain minimal functionality for the purposes of booting and repairing a system.

sbin is short for system binary. Similar to bin, it is also a place for storing executable programs. But these executable programs are essential for system configuration, maintenance and administrative tasks.

/usr directory

usr stands for Unix System Resources. It belongs to the user applications as opposed to /bin or /sbin directories which belong to system applications. Any application installed here is considered nonessential for basic system operation. However, this is one of the most important directories in the system as it contains all the user-level binaries, their documentation, libraries, header files, etc. This directory is read-only and applications should not write anything into it.

/etc directory

This is where all your system-wide configurations are stored. So, if you have a system-wide application on your Linux, you can find its configuration files here.

/var directory

var stands for variables. This directory contains variable data like system logging files, mail and printer spool directories, and transient and temporary files. These are typically file and directories that are expected to grow in size. For example, /var/crash holds information about every time a process has crashed. Or /var/log has all log files for your computer and its applications, which grow constantly with time.

/boot directory

This is where your boot loader lives. It contains the static bootloader, kernel executable and configuration files required to boot a computer.

/sys directory

This is where you can interact with the kernel. In other words, you can consider it as an interface to the kernel. This directory is a virtual filesystem, which means the files live on memory and disappear after shutdown.

67. Explain absolute and Relative Paths

Answer: An absolute path is a path that describes the location of a file or folder regardless of the current working directory; in fact, it is relative to the root directory. It contains the complete location of a file or directory, hence the name.

A relative path is a path that describes the location of a file or folder in relative to the current working directory.

68. What are the different ways of creating a File in linux System? Write an example of each and the difference between them.

Answer: There are mainly four ways of creating files in Linux. All of them have their own purpose and benefits.

1. cat command

It is the most universal command/tool for creating files on Linux systems. We cannot edit a file using the cat command.To create files and write the data into them

cat >file1

2. touch command

We can create an empty file (or multiple empty files) using this command. But its main purpose is to change or update the time-stamp of a file

Creating a file

touch file1

3. vi command

Its main function is to edit files. It is commonly used by programmers to edit the textual content of any file on vi text editor. Major operations that can be done using it are as follows:

Note: To save and exit from the vi text editor, press the Escape key and then type :wq and hit enter.

Create a file

vi file_1

4. nano command

It may/may not be found in all distributions of LINUX. We can create as well as edit files.

Create using nano file_1 then write the content then exit using ctrl+o and ctrl+X.

69. In how many ways we can delete the files from linux system? write an example of each and teh difference between them.

Answer: There are different ways to delete the files from linux system which are as follows.

1. Remove file by using "unlink":

We may use the unlink command to permanently delete a single file.

$ unlink {file-name}

2. Delete a single file:

The rm command, which facilitates deleting one or more files simultaneously, is a more widely used command for removing files.

$ rm {file-name}

Using the "-i" flag to force rm to prompt for confirmation before removing a file:

$ rm -i {file-name}

Using the -f flag to remove write-protected files without asking for clarification.

```
$ rm -f {file-name}
```

3. Multiple files can be deleted:

Bypassing multiple filenames as arguments to rm, you can delete multiple files.

```
$ rm {file-name-1} {file-name-2} {file-name-3} ...
```

4. Delete the archive:

The rm command with the -d flag can be used to remove an empty directory.

```
$ rm -d {dir-name}
```

70. Archiving files using linux command, write a command to archive set of files from linux commands.

Answer: 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.

```
$ tar cvf file.tar *.c
```

71. Extract the archived files from the above step.

Answer: To extract an archive, use the tar -xf command followed by the archive name, and to create a new one use tar -czf followed by the archive name and the files and directories you want to add to the archive.