Basic Unix Command Line Assignment 1:

Assignments:-

1. List the contents of a directory and their attributes



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



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

**ls -s -h**



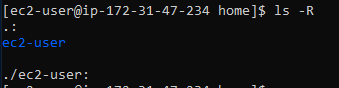
1. Show all files and folders including hidden one

**ls -a**



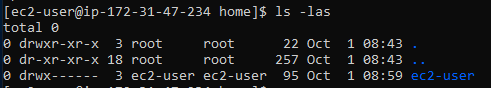
1. List directories recursively

**ls -r**



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

**ls -laS**



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

**ls -lt**

1. Display the location of a program/command, where it is installed.

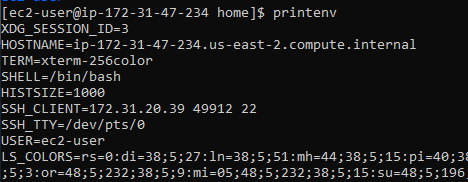
**which**

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

**cd**

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

**printenv**



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

**We can get all environmental variables**

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

**echo**

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

**echo $PATH**



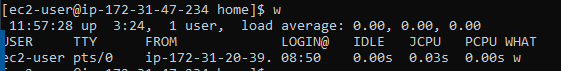
1. Is linux a case-sensitive operating system?

**Yes**

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

**Yes**

1. Display your currently logged in user

**whoami**

1. how do you change the currently logged in user to another user?

**su –[username]**

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

**chsh**

1. How do you reboot the system?

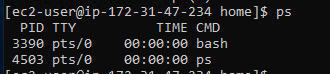
**init 6**

1. How do you shutdown the system?

**init 0**

1. Display all the major running processes in the system

**ps -a**



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

PPID - The parent process's ID.

TTY - The terminal from which the process (or its parent) was started. A question mark indicates there is no controlling terminal.

TIME - The total amount of CPU time used by the process since it began.

CMD - The command that generated the process.

1. Display the name of the system kernel

**uname -s**

1. display the kernel release number

**uname -v**

1. display teh machine type of the current kernel

**uname -i**

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

**uname -o**

1. Display all info that uname command can show.

**uname -a**

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

**pwd**

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

**cd absolute path**

1. Go up one directory

**cd ..**

1. Return to last directory

**cd**

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

**cd ~username**

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

**history**

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

**~/.bash\_history**

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

**500**

1. How can you modify bash's history behaviour

To customize bash command history, we will have to make changes in the ~/.bashrc file. To edit the ~/.bashrc file, use the following command:

**nano ~/.bashrc**

Once done with modifying the file, use Ctrl+O and Ctrl+X to save and close the nano editor. Then run the following command to apply the modifications.

**source ~/.bashrc**

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

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 ;-)

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

* The C Shell : It includes helpful programming features like built-in arithmetic and C like expression syntax.

Denoted as **csh**

* The Bourne Shell : It is faster and more preferred. It lacks features for interactive use like the ability to recall previous commands.

Denoted as **sh**

* The Korn Shell : It includes features like built-in arithmetic and C like arrays, functions and string-manipulation facilities. It is faster than C shell, it is compatible with script written for C shell.

It is denoted as **ksh**

* GNU Bourne-Again Shell : It is compatible to the Bourne shell. It includes features from Korn and Bourne shell.

Denoted as **bash**

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

**login shell**: A login shell is the first process that executes under your user ID when you log in for an interactive session

**non-login shell**: When you log in on a text console, or through SSH, or with su -, you get an interactive login shell

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

**loginshell**=yes : Place in .profile file

**Non loginshell** : These shells don't read .login or .profile. In addition, bash allows a nonlogin shell to read ~/.bashrc or not

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

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 ~/.profile, in that order, and reads and executes commands from the first one that exists and is readable

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

no login means only prices can be run

While shell login ID disabled. Only files with full perm and own perm can be read

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

The /etc/profile file – it stores system-wide environment configurations and startup programs for login setup. All configurations that you want to apply to all system users’ environments should be added in this file.

1. Explain the Order of file usage from the system/user's home directory when user logs in to the System.
2. What are Shell Variables, list major shell variables and what do they represent?

A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

A variable is nothing more than a pointer to the actual data. The shell enables you to create, assign, and delete variables

1.Defining Variables

2.Accessing Values

3.Read-only Variables

4.Unsetting Variables

1. How we see all our env variables?

**printenv**

**env**

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

**env | sort**

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

Variables have following format

KEY=value

KEY="Some other value"

KEY=value1:value2

1. How do you create your own variable?

myname=’name’

number=10

1. How do you start a new bash shell?

bash -c 'gnome-terminal -x cd /absolute-path && program\_nam

1. Difference between Local/Shell variables to Global Variable

Global variables are declared outside any function, and they can be accessed on any function in the program.

Local variables are declared inside a function, and can be used only inside that function.

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

Globbing is the act of locating files on a filesystem using one or more globs. The src() method expects a single glob string or an array of globs to determine which files your pipeline will operate on.

1. List all entries with extension ".sh"

**ls –l \*.sh**

1. List all entries with numbers in it.

**ls -l [1-5] : you can give the set of range**

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

**s -l {?????.sh,\*st.txt}, ls a\*+(5|7)**

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

**s -l {?????.sh}**

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

Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to disable special treatment for special characters, to prevent reserved words from being recognized as such, and to prevent parameter expansion.

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

Hello

./test.sh: line 3: Word: command not found

shell returned 127

Let us now try using a quoted character −

#!/bin/sh

echo Hello\; Word

Upon execution, you will receive the following result

Hello; Word

1. How do you find a particular files/directories based on a particular search criteria? HINT:- look for commands -> locate, find and whereis

Find . test.txt

find / -type d -name assignment: Search on directory

locate "\*.html" -n 20: show the first 20 files that end with 'html'

locate -i \*text.txt\*: Ignore case sensitive

whereis ls: search for ls command

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

**whereis**: will search only particular paths to find binaries and or manpages. The manpages tells you where whereis looks.

**locate**: locate uses a database created by an updatedb to efficiently locate files. Works great, assuming your database is updated often enough to be reasonable upto date. Most boxes using locate have the updatedb occuring in cron.

**find**: find is perhaps one of the most powerful commands there is. For just locating a file/program of a particular name, it'll definitely be slower than locate or whereis becuase it will search each and every path recursively from it's start point.

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

Globbing is find files based on charecters and numbers only.when it comes to locate, find and whereis find files by time, location, permissions, date, etc.

1. Explain the Linux File System.

Linux file system is generally a built-in layer of a Linux operating system used to handle the data management of the storage. It helps to arrange the file on the disk storage. It manages the file name, file size, creation date, and much more information about a file.

1. Explain absolute and Relative Paths

An absolute or full path points to the same location in a file system, regardless of the current working directory. To do that, it must include the root directory. By contrast, a relative path starts from some given working directory, avoiding the need to provide the full absolute path.

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

* Create a File with Touch Command

**touch test.txt**

* Create a New File with the Redirect Operator

**> test2.txt**

* Create File with cat Command

**cat > test3.txt**

* Create File with echo Command

**echo ‘Random sample text’ > test4.txt**

* Create File with printf Command

**printf ‘First line of text\n’ test5.txt**

* Using Text Editors
* Vi Text Editor

**vi test7.txt**

* Vim Text Editor

**vim test8.txt**

* Nano Text Editor

**nano test9.txt**

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

Remove or delete a file.

**rm linuxstufff.log**

Delete multiple files at once.

**rm file1.txt file2.txt file3.txt file4.txt**

Delete the files interactively.

**rm -i linuxstufff.log**

Delete an empty directory in linux

**rm -d appdata/**

**rmdir appdata**

Deleting a directory recursively using ‘-r’ option

**rm -r dbstore/**

Delete the files and sub-directories interactively.

**rm -ri dbstore/**

Deleting files forcefully using ‘-f’ option

**rm -f tech.tx**

Prompt once before deleting more than three files or recursive delete.

**rm -I linux\_store/app\***

Regular expression in rm command

**rm -f \*.txt**

Delete large number files using rm command.

**rm \*.log**

Delete a file which starts with hyphen symbol (-)

**rm -- \ -store**

**rm ./\ -store**

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

**tar cf archive.tar file1 file2 file3**

1. Extract the archived files from the above step.

**tar xf archive.tar**