**Day\_0\_Shell\_Script\_Assignment.sh**

1. **Write a shell script which will execute following set of tasks:**
   1. **Create a folder named ‘sample’ in your ‘home’ directory**

-> ~mkdir sample

* 1. **Inside ‘sample’ folder, create a file called ‘sample.txt’**

-> cd sample

-> touch sample.txt

* 1. **Add the following content to the file:  
      Hi! This is just a sample text file created using shell script.**

**-**> sample echo "This is the sample script created using shell script." > sample.txt

* 1. **Print the contents of the file.**

-> sample cat sample.txt

* 1. **Print the number of occurrences of letter ‘t’ in ‘sample.txt’**

-> sample grep -o "t" sample.txt | wc -l

* 1. **Change the owner permissions to allow all the operations on the file. ( Read, Write, Execute )**

-> sample chmod u+rwx sample.txt

* 1. **Write command to append following content in sample.txt file:**

**Hi! This is just another sample text added to the file.**

-> sample echo "Hi! This is just another sample text added to the file." >> sample.txt

1. **Change the group permissions to allow only read operation.**

-> sample chmod g-wx sample.txt

1. **Change the all users permission to deny any sort of access to ‘sample.txt’**

-> sample chmod 000 sample.txt

1. **Write command to create file named sample2.txt with content similar to that of sample.txt**

-> sample chmod u+r sample.txt

-> sample cp sample.txt sample2.txt

1. **Add some random 1000 lines in the sample.txt file.**

Using a loop we can add random 1000 lines in the sample.txt. This loop generates 1000 random numbers between 0 and 999,999 (inclusive) and appends each as a new line to sample.txt.

**e.g.**

-> for i in {1..1000}; do

echo $(( RANDOM % 1000000 )) >> sample.txt

done

1. **Write command to print top 50 lines of file**

-> sample head -n 50 sample.txt

1. **Write command to print bottom 50 lines of the file**

-> sample tail -n 50 sample.txt

1. **Add 5 files in the same folder named: prog1.txt, prog2.txt, program.txt, code.txt, info.txt**

-> sample touch prog1.txt prog2.txt program.txt code.txt info.txt

1. **Write the command to list files which has “prog” in its name**

-> sample ls \*prog\*

1. **Create an alias of the command used at step o. Such that following command:**

**list prog**

**Should have the same output as of command at step o.**

-> sample alias listprog=’ls \*prog\*’

1. **Create two files “a.txt” and “b.txt”. Write a command to get the difference between the contents in two files.**

* Using vim editor we can see the difference between a.txt and b.txt clearly

-> sample touch a.txt b.txt

-> sample vimdiff a.txt b.txt

Or

-> sample vim -d a.txt b.txt

1. **What is the difference between source and sh commands?**

* **sh commands:  
  sh** basically stands for shell commands. It is used to invoke a shell interpreter to execute scripts & commands. The sh utility acts as a bridge between user and the os. Historically **sh** refers to the Bourne shell. Nowadays we use modern shell bash.

Syntax of sh command:  
sh [options] [script\_file] [arguments]

Here's a breakdown of the components −  
**[options]** − Flags that modify the behavior of the shell.  
**[script\_file]** − The script you want to run using the shell interpreter.  
**[arguments]** − Additional inputs required by the script.  
  
Example of sh command:  
Running a shell script-  
E.g. sh [script](http://script.sh).sh

* **source**

1. The 'source' command is a built-in feature of the shell, designed to execute commands stored within a file directly in the current shell environment.
2. When you use the 'source' command, it reads the contents of the specified file, typically a sequence of commands, and executes them as if they were typed directly into the terminal.

Syntax of source command in LINUX:

source FILE\_NAME

Examples of source command in LINUX:

1. Execution of Shell Scripts.
2. Modifying Environment Variables.
3. Loading Configuration Files.
4. Defining Configuration Functions.
5. **What is the difference between ls and lsof?**

* **ls->** It is basically used to display all the content of the directories. It shows the information about all the files, subdirectories under that specific directory.

We can also use some options with ls to modify its behaviour:

I. -l -> It basically lists all the files & subdirectories within that specific directory with the detailed information like permission, owner, size.

II. -a -> This option is used to list all the files including hidden files inside that

Directory.

III. -la -> It is a combination of the above two options which list all the files, subdirectories including hidden files with the information like permissions, owner and size.

* **lsof->** Basically lsof stands for “list open files” . It allows users to gain insights into the currently opened files on the system. Basically, it gives the information to find out the files which are opened by which process and information about which port they are using and many more things. With one go it lists out all open files in the output console.   
    
  We can also use some options with lsof to modify its behaviour:

I. -c -> It lists files opened by a specific process name.

II. -u -> It lists files opened by a specific user.

III. -p -> It lists files opened by a specific process ID.

IV. -D -> Its lists files opened by a specific directory

1. **Create directories ./hello/world (World dir is inside hello dir) using mkdir command where neither hello or world exists. It should be a single command without the use of &&.**

-> mkdir -p ./hello/world

Here -p refers to the parent if no such parent exists then it first creates that parent and then create the subdirectory inside it

1. **How can you permanently set an environment variable using a SHELL terminal?**

* First i need to create an environment variable  
  E.g. export TEST\_VAR= ‘test’
* Here i created an TEST\_VAR environment variable
* If, i need to print my env so i can use following command  
  E.g. printenv TEST\_VAR
* Now, if i need to set that environment variable permanently in shell terminal for that i need to store that environment variable in a **.bashrc** file.
* For that i use nano text editor  
  E.g. nano .bashrc  
  By, above command the hidden file .bashrc is opened in the nano editor. Now I need to store that env in that file.
* Now to add env i need to run the following command:  
  E.g. export TEST\_VAR=’test’
* Now in this way we manipulate the .bashrc file now to save changes in the .bashrc file we have to run following command:  
  E.g. source .bashrc
* By using following command we can easily check our env is stored permanently in the shell terminal or not   
  E.g. env   
  This command list all the environment variables in the shell terminal

**Note:** I am also providing the screenshots of my hands on for all the above shell scripts for reference :

