SHELL SCRIPTING INTERVIEW QUESTIONS

A shell can also take commands as input from file, we can write these commands in a file and can execute them in shell to avoid this repetitive work. These files are called Shell Scripts Each shell script is saved with `.sh` file extension e.g., myscript.sh.

1.Write some advantages of shell scripting.

Ans: Shell scripting is primarily used to automate repetitive system tasks, such as backing up files, monitoring system resources, and managing user accounts. By turning a series of commands into a script, system administrators can save time, increase accuracy, and simplify complex tasks.

Shell scripts are easy to use and quicker to write.

2. Write some limitations of shell scripting.

Ans: Errors are frequent and costly, and a single error can alter the command.

The execution speed is slow.

3. Name the file in which shell programs are stored.

A file called sh stores shell programs. Sh files contain commands written in a scripting language that is run by Unix shells.

4.what is a variable in shell?

Ans: Variable is a character String to which we will assign a value. Value could be Number,Text,Filename,device or any other type of data.

EX: variable ="Hello"
echo $variable

5.What are the different types of variables in shell scripting?

Ans: Shell scripts usually have two types of variables:

- **System-defined variables:** Also called environment variables, these are special built-in variables in the Linux kernel for each shell. They are normally defined in capital letters by the OS (Linux) and are standard variables.
  Example:
  SHELL
  It is a Unix Defined or System Variable, which specifies the default working shell.
- **User-defined variables:** These variables are created and defined by users in order to store, access, read, and manipulate data. In general, they are defined in lowercase letters. The Echo command allows you to view them.
  Example:

$ a=10

In this case, the user has defined the variable ‘a’ and assigned it the value 10.

6. What are control instructions?

Control instructions specify how the different instructions will be executed in the script. They are primarily used to determine the control flow in Shell programs.

In shell programs, control instructions govern how execution flows.

7. What is the shebang line in shell scripting?

Ans: The shebang is the combination of the # (pound key) and ! (exclamation mark). This character combination is used in the very first line of the script. It is used to specify the interpreter with which the given script will be run by default.

Ex: #!/bin/bash

8. Name the alternative command for echo.

Ans: The tput command is an alternative command for echo.

9. What is the importance of $#?

$# usually holds the number of arguments, although it may be different for functions. To put it simply, it was used to store the number of command-line arguments passed to a shell script.

10. Write the command that is used to execute a shell file.

Ans: you can execute shell script by:

sh script-name-here.sh

11. Name the command that can be used to find out the number of arguments passed to the script?

The following command will display the number of arguments passed to the script: **echo $ #**

12. How long does a variable in a shell script last?

Variables inside shell scripts have a lifespan of only as long as the execution lasts.

13. What does the . (dot) indicate at the beginning of the file name? How can it be listed?

If the file name begins with ".", it is a hidden file. When a dot appears at the beginning of a filename, the file is hidden in most file managers and shell programs.

The Is command must be run with the **"–a"** flag if you wish to see hidden files.

14. What is the use of the "$?" command?

By using the command "$?", you can find out the status of the last command executed.

15. What is the best way to run a script in the background?

For a script to run in the background, simply add "&" at the end of the command.

Example:

script.sh &

command &

16. What is the difference between $* and $@ in Linux?

Ans: "$@" expands to a string for each command line argument.

"$*" expands into a single string with all of the words.

Without quotes both expand to a string per word.

17. How will you pass and access arguments to a script in Linux?

In scripts, arguments are passed as follows:

scriptName "Arg1" "Arg2"…."Argn"

Arguments in a script can be accessed as follows:

$1 , $2 .. $n

18. Write the difference between "=" and "==".

- **= operator:** Assigning the value into a variable is accomplished by using the = operator. It is referred to as the assignment operator.
  Example: Suppose variable a holds 5 and variable b holds 2, then:

```
a = $b;          #Would assign value of b to a
```

- **== operator:** This is used to compare strings. In the double equals operator, both operands are compared. If they are equal, it returns true, otherwise, it returns false.
  Example: Suppose variable a holds 5 and variable b holds 2, then:

```
[ $a == $b ];     #Comparing the values of a and b.
```

19. Name the command that is used to display the shell's environment variable.

Shell functions, along with other Linux programs, are controlled by environmental variables. Any child process of the shell has access to an environment variable. These variables are necessary for some programs to function properly. **env** or **printenv** commands can be used to display the shell's environment variables.

20. Name different commands that are available to check the disk usage.

The following commands are available to check disk space usage:

- **df** – It is used to find out how much space is left on a disk.

- **du** – With this command, you can find out how much disk space the specified files and each subdirectory take up.
- **dfspace** – Using this command, you can check the amount of free disk space in MB

19. How will you find total shells available in your system?

Within your system, there are several shells available. If you want to find all the shells on the system, you can use $ cat /etc/shells.

Example:

$ cat /etc/shells

Output:

/bin/sh

/bin/bash

/sbin/nologin

/bin/ksh

/bin/csh

20. Name the command that one should use to know how long the system has been running.

Using the Uptime command, you can see how long your system has been active.

Example: $ uptime

21. Write the difference between $$ and $!?

You can use $$ to get the process id of the current process.

However, $! displays the process id for a background process that recently went away

22. How to use pipe commands?

Pipe command allows you to use several commands in the same way, in which the output of one is used as input for another. Like a pipeline, each process output is directly input to the next one. A pipe is represented by the symbol "|". The flow of data through a pipeline is unidirectional, i.e., from left to right.

Syntax :

command_1 | command_2 | command_3 | .... | command_N