UNIX SHELL SCRIPTING INTERVIEW QUESTIONS

Q #1) What is Shell?

Ans: Shell is a command interpreter, which interprets the command which the user gives to the kernel. It can also be defined as an interface between a user and operating system.

Q #2) What is Shell Scripting?

Ans: Shell scripting is nothing but series or sequence of UNIX commands written in a plain text file. Instead of specifying one job/command at a time, in shell scripting we give a list of UNIX commands like a to-do list in a file to execute it.

Q #3) What is the Importance of writing Shell Scripts?

Ans: The points given below explain the importance of writing shell scripts.

- Shell script takes input from the user, file and displays it on the screen.
- Shell scripting is very useful in creating your own commands.
- It is helpful in automating some tasks of the day to day life.
- It is useful for automating system administration tasks.
- Mainly it saves time.

Q #4) List some of the common and most widely used UNIX commands.
Ans: Given below is a list of widely used UNIX Commands.

Command	Example/Usage of Command	Description
ls	1. $ ls 2. $ ls –lrt or $ ls -ltr	1. It lists files in the current directory. 2. It lists files in the long format.
cd	1. $ cd 2. $ cd test 3. $ cd .. (after cd space needs to be given before entering two dots.)	1. It changes directory to your home directory. 2. It changes directory to test. 3. It moves back to one directory or to the parent directory of your current directory.
mkdir	$ mkdir test	It creates a directory called test.
rmdir	$ rmdir test1 **CAUTION: Be careful while using this command.**	It removes directory test1.
cp	1. $ cp file1 test 2. $ cp file1 file1.bak	1. It copies file1 to test directory. 2. It takes backup of file1.
rm	$ rm file1 **CAUTION: Be careful while using this command.**	It removes or deletes a file1.
mv	$ mv file1 file2	It moves or renames file1 to file2.
more	$ more	It checks or display one page at a time.
touch	$ touch test	It creates an empty file called test.
cat	1. $ cat File1 2. $ cat test1 > test2	1. It displays contents of File1. 2. It creates a new file test2 with the contents of test1.
compress	$ compress file1	It reduces the size of file1 and creates a compressed file called file1.z and deletes file1.
date	$ date e.g. Output: Tuesday, September 12, 2017 06:58:06 AM MDT	It displays current date and time.
diff	$ diff file1 file2	It displays line by line difference between file1 and file2.
find	$ find . –name ‘*.t’ -print	It searches in the current directory and in all its subdirectories for files ending with .t, and writes their names in the output.
finger	$ finger	It displays information about user.
who	$ who	It lists the users those who are logged in on the machine.
grep	1.$ grep Hello file1 2.$ grep –c Hello file1	1. It searches for the lines containing Hello in file1. 2. It gives count or number of lines that contains Hello in file1.

Command	Example/Usage of Command	Description
kill	kill $ kill 1498	It kills the process which is having PID as 1498.
lpr	1.$ lpr –Pprinter1 test 2.$ lp file1	1. It sends file test to print it on printer1. 2. It prints file1.
man	$ man ls	It displays online manual or help about ls command.
passwd	$ passwd	It is used to change the password.
pwd	$ pwd e.g. Output: /u/user1/Shell_Scripts_2017	It displays present working directory.
ps	$ ps e.g. Output: PID TTY TIME COMMAND 1498 3b 0:10 sh 1500 3b 0:05 sh	It displays the list of processes which are currently running on the machine.
talk	$ talk user1	It is used to talk to the user1 who is currently logged into the same machine.
wc	$ wc file1 e.g. Output: 4 6 42 file1	It counts the number of lines, words and characters in file1.
chmod	$ chmod 744 file1	It changes the permissions of file1 & assigns this permission rwxr--r--
gzip	$ gzip file1	It compresses the file1. After compression file1 should look like this, file1.gz
gunzip	$ gunzip file1.gz	It uncompresses the file1.gz. After uncompression file1.gz should look like this, file1
history	$ history	It lists all the commands which are recently used.
logname	$ logname e.g. Output: user1	It prints log name of the user.
uname	$ uname e.g. Output: SunOS	It gives information about unix system which you are using.
tty	$ tty e.g. Output: /dev/pts/1	It displays the device name of your terminal.
sort	$ sort file1	This will sort the contents of file1 and displays sorted output on the screen.
head	$ head -15 file1	It displays first 15 lines of the file.
tail	$ tail -15 file1	It displays last 15 lines of the file.

Q #5) Shell programs are stored in which file?
Ans: Shell programs are stored in a file called **sh**.
Q #6) What are the different Types of Shells available?
Ans: There are mainly 4 important types of shells that are widely used.
And they include:

- Bourne Shell (sh)
- C Shell (csh)
- Korn Shell (ksh)
- Bourne Again Shell (bash)

Q #7) What are the Advantages of C Shell over Bourne Shell?
Ans: The advantages of C Shell over Bourne Shell are:

- C shell allows aliasing of commands i.e. a user can give any name of his choice to the command. This feature is mainly useful when a user has to type the lengthy command again and again. At that point of time, instead of typing a lengthy command a user can type the name that he has given.
- C shell provides command history feature. C shell remembers the previously typed command. Thus, it avoids typing the command again and again.

Q #8) In a typical UNIX environment how many kernels and shells are available?
Ans: In a typical UNIX environment, only one kernel and many shells are available.
Q #9) Is separate compiler required for executing a shell program?
Ans: A separate compiler is not required to execute a shell program. The shell itself interprets the command in the shell program and executes them.
Q #10) How many shell scripts come with UNIX operating system?
Ans: There are approximately 280 shell scripts that come with the UNIX operating system.
Q #11) When should shell programming/scripting not be used?
Ans: Generally, shell programming/scripting should not be used in the below instances.

- When the task is very much complex like writing the entire payroll processing system.
- Where there is a high degree of productivity required.
- When it needs or involves different software tools.

Q #12) Basis of shell program relies on what fact?
Ans: The basis of shell programming relies on the fact that UNIX shell can accept commands not just only from the keyboard but also from a file.
Q #13) What are the default permissions of a file when it is created?
Ans: 666 i.e. rw-rw-rw- is the default permission of a file when it is created.
Q #14) What can be used to modify File permissions?
Ans: File permissions can be modified using **umask**.
Q #15) How to accomplish any task via shell script?
Ans: Any task can be accomplished via shell script at the dollar ($) prompt and vice versa.
Q #16) What are Shell Variables?
Ans: Shell variables are the main part of shell programming or scripting. They mainly provide the ability to store and manipulate information within a shell program.
Q #17) What are the two types of Shell Variables? Explain in brief.
Ans: The two types of shell variables are:
#1) Unix Defined Variables or System Variables – These are standard or shell defined variables. Generally, they are defined in CAPITAL letters.
Example: SHELL – This is a Unix Defined or System Variable, which defines the name of the default working shell.
#2) User Defined Variables – These are defined by users. Generally, they are defined in lower letters
Example: $ a=10 –Here the user has defined a variable called '**a**' and assigned value to it as 10.
Q #18) How are shell variables stored? Explain with a simple example.
Ans: Shell variables are stored as string variables.
Example: $ a=10
In the above statement a=10, the 10 stored in '**a**' is not treated as a number, but as a string of characters 1 and 0.
Q #19) What is the lifespan of a variable inside a shell script?
Ans: The lifespan of a variable inside shell script is only until the end of execution.
Q #20) How to make variables as unchangeable?
Ans: Variables can be made unchangeable using **readonly**. For instance, if we want variable **a** value to remain as **10** and not to be changed then we can achieve this using **readonly**.
Example:

```
$ a=10

$ readonly a
```

Q #21) How variables can be wiped out?
Ans: Variables can be wiped out or erased using the **unset** command.
Example:
$ a =20

$ unset a

Upon using the above command the variable '**a**' and its value **20** get erased from shell's memory.
CAUTION: Be careful while using this **unset** command.
Q #22) What are positional parameters? Explain with an example.
Ans: Positional parameters are the variables defined by a shell. And they are used whenever we need to convey information to the program. And this can be done by specifying arguments at the command line.
There are totally 9 positional parameters present i.e. from $1 to $9.

Example: $ Test Indian IT Industry has grown very much faster
In the above statement, positional parameters are assigned like this.

$0 -> Test (Name of a shell program/script)

$1 ->Indian

$2 -> IT and so on.

Q #23) What does the. (dot) indicate at the beginning of a file name and how should it be listed?
Ans: A file name which begins with a .(dot) is called as a hidden file. Whenever we try to list the files it will list all the files except hidden file.
But it will be present in the directory. And to list the hidden file we need to use –a option of ls. i.e. $ ls –a.

Q #24) Generally, each block in UNIX is how many bytes?
Ans: Generally, each block in UNIX is of 1024 bytes.
Q #25) By default, a new file and a new directory which is being created will have how many links?
Ans: New file contains **one** link. And a **new directory** contains **two** links.
Q #26) Explain about file permissions.
Ans: There are 3 types of file permissions as shown below:

Permissions	Weight
r – read	4
w – write	2
x - execute	1

The above permissions are mainly assigned to owner, group and to others i.e. outside the group. Out of 9 characters first set of 3 characters decides/indicates the permissions which are held by the owner of a file. The next set of 3 characters indicates the permissions for the other users in the group to which the file owner belongs to.

And the last 3 set of characters indicate the permissions for the users who are outside the group. Out of the 3 characters belonging to each set, the first character indicates the "read" permission, the second character indicates "write" permission and the last character indicates "execute" permission.

Example: $ chmod 744 file1
This will assign the permission rwxr–r–to file1.

Q #27) What is a file system?
Ans: The file system is a collection of files which contain related information of the files.

Q #28) What are the different blocks of a file system? Explain in brief.
Ans: Given below are the main 4 different blocks available on a file system.

File System	
Block No.	Name of the Block
1st Block	Boot Block
2nd Block	Super Block
3rd Block	Inode Table
4th Block	Data Block

Super Block: This block mainly tells about a state of the file system like how big it is, maximum how many files can be accommodated etc.
Boot Block: This represents the beginning of a file system. It contains bootstrap loader program, which gets executed when we boot the host machine.
Inode Table: As we know all the entities in a UNIX are treated as files. So, the information related to these files are stored in an Inode table.
Data Block: This block contains the actual file contents.

Q #29) What are the three different security provisions provided by UNIX for a file or data?
Ans: Three different security provisions provided by UNIX for a file or data are:

- It provides a unique user id and password to the user, so that unknown or unauthorized person should not be able to access it.
- At file level, it provides security by providing read, write & execute permissions for accessing the files.
- Lastly, it provides security using file encryption. This method allows encoding a file in an unreadable format. Even if someone succeeds in opening a file, but they cannot read its contents until and unless it is decrypted

Q #30) What are the three editors available in almost all the versions of UNIX?
Ans: The three editors are ed, ex & vi.

Q #31) What are the three modes of operation of vi editor? Explain in brief.
Ans: The three modes of operation of **vi editors** are,
(i) Command Mode: In this mode, all the keys pressed by a user are interpreted as editor commands.
(ii) Insert Mode: This mode allows for insertion of a new text and editing of an existing text etc.
(iii) The ex-command Mode: This mode allows a user to enter the commands at a command line.

Q #32) What is the alternative command available to echo and what does it do?
Ans: tput is an alternative command to **echo**.
Using this, we can control the way in which the output is displayed on the screen.

Q #33) How to find out the number of arguments passed to the script?
Ans: The number of arguments passed to the script can be found as shown below.
echo $ #

Q #34) What are control instructions and how many types of control instructions are available in a shell? Explain in brief.
Ans: Control Instructions are the ones, which enable us to specify the order in which the various instructions in a program/script are to be executed by the computer. Basically, they determine a flow of control in a program.
There are 4 types of control instructions that are available in a shell.

- **Sequence Control Instruction** – This ensures that the instructions are executed in the same order in which they appear in the program.
- **Selection or Decision Control Instruction** – It allows the computer to take a decision as to which instruction is to be executed next.
- **Repetition or Loop Control Instruction** – It helps a computer to execute a group of statements repeatedly.
- **Case-Control Instruction** – This is used when we need to select from several alternatives.

Q #35) What are Loops and explain three different methods of loops in brief?

Ans: Loops are the ones, which involve repeating some portion of the program/script either a specified number of times or until a particular condition is being satisfied.

3 methods of loops are:

- **For** loop – This is the most commonly used loop. For loop allows specifying a list of values which the control variable in the loop can take. The loop is then executed for each value mentioned in the list.
- **While** loop – This is used in a program when we want to do something for a fixed number of times. While loop gets executed till it returns a zero value.
- **Until** loop – This is similar to while loop except that the loop executes until the condition is true. Until loop gets executed at least once till it returns a non-zero value.

Q #36) What is IFS?

Ans: IFS stands for **Internal Field Separator**. And it is one of the system variables. By default, its value is space, tab, and a new line.

It signifies that in a line where one field or word ends and another begins.

Q #37) What is a Break statement and what is it used for?

Ans: The break is a keyword and is used whenever we want to jump out of a loop instantly without waiting to get back to the control command.

When the keyword break is encountered inside any loop in the program, control will get passed automatically to the first statement after a loop. A break is generally associated with an if.

Q #38) What is Continue statement and what is it used for?

Ans: Continue is a keyword and is used whenever we want to take the control to the beginning of the loop, by passing the statements inside the loop which have not yet been executed.

When the keyword continue is encountered inside any loop in the program, control automatically passes to the beginning of a loop. Continue is generally associated with an if.

Q #39) What are Metacharacters in a shell? Explain with some examples.

Ans: Metacharacters are special characters in a program or data field which provides information about other characters. They are also called as, regular expressions in a shell.

Example:

ls s* – It lists all the files beginning with character 's'.

Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts2017> ls s*
```

Output:

```
script1   script2
```

$ cat script1 > script2 – Here output of cat command or script1 will go to a script2.

Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts2017> cat script1 > script2
```

Output:

```
#!/bin/bash
# script1
# Usage: script1
echo Hello !!
echo How are you?
script2  #here it calls the script2
pwd
```

$ ls; who – This will execute ls first and then who.

Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts2017> ls; who
```

Output:

```
script10                script5
Script2                 script6
```

```
tibadm      pts/1        Sep 14 08:22
crmadm      pts/2        Sep 14 08:43
```

Q #40) How to execute multiple scripts? Explain with an example.

Ans: In a shell, we can easily execute multiple scripts i.e. one script can be called from the other. What we have to do is, we need to mention the name of a script to be called when we want to invoke it.

Example: In the below program/script upon executing the first two echo statements of script1, shell script executes script2. Once after executing script2, the control comes back to script1 which executes a **pwd** command and then terminates.

Code for script1

```
#!/bin/bash
# script1
# Usage: script1
echo Hello !!
echo How are you?
script2  #here it calls the script2
pwd
~
```

Code for script2

```
#script2
echo Software testing is an interesting job.
~
```

Execution of script1 over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts2017> script1
```

Output displayed on the Editor upon executing script1

```
Hello !!
How are you?
Software testing is an interesting job.
/u/user1/Shell_Scripts_2017
~
```

Q #41) Which command needs to be used to know how long the system has been running?
Ans: uptime command needs to be used to know how long the system has been running.
Example: $ uptime
Upon entering the above command at shell prompt i.e. $ uptime, the output should look something like this.

9:21am up 86 day(s), 11:46, 3 users, load average: 2.24, 2.18, 2.16

Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts_2017> uptime
```

Output:

```
9:21am  up 86 day(s), 11:46,  3 users,  load average: 2.24, 2.18, 2.16
```

#42) How to find the current shell which you are using?
Ans: We can find the current shell what we are using with echo $SHELL.
Example: $ echo $SHELL
Execution over Shell Interpreter/Editor

```
$ echo $SHELL
```

Output:

```
/bin/bash
```

Q #43) How to find all the available shells in your system?
Ans: We can find all the available shells in our system with $ cat /etc/shells.
Example: $ cat /etc/shells
Execution over Shell Interpreter/Editor

```
$ cat /etc/shells
```

Output:

```
/bin/sh
/bin/bash
/sbin/nologin
/bin/ksh
/bin/dash
/bin/tcsh
/bin/csh
```

Q #44) How to read keyboard inputs in shell scripts?
Ans: Keyboard inputs can be read in shell scripts as shown below,
Script/Code

```
#!/bin/bash
#script6
read name
echo "Hello $name"
```

Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts_2017> script6
```

Output:

```
Mahesh
Hello Mahesh
```

Q #45) How many fields are present in a crontab file and what does each field specify?
Ans: The **crontab** file has six fields. The first five fields tell **cron** when to execute the command: minute(0-59), hour(0-23), day(1-31), month(1-12), and day of the week(0-6, Sunday = 0).
And the sixth field contains the command to be executed.

Q #46) What are the two files of crontab command?
Ans: Two files of crontab command are:

- cron.allow – It decides which users need to be permitted from using crontab command.
- cron.deny – It decides which users need to be prevented from using crontab command.

Q #47) What command needs to be used to take the backup?
Ans: tar is the command which needs to be used to take the backup. It stands for tape archive. The **tar** command is mainly used to save and restore files to and from an archive medium like tape.
Q #48) What are the different commands available to check the disk usage?
Ans: There are three different commands available to check the disk usage.
And they are:
df – This command is used to check the free disk space.
du – This command is used to check the directory wise disk usage.
dfspace – This command is used to check the free disk space in terms of MB.
Q #49) What are the different communication commands available in Unix/shell?
Ans: Basically, there are 4 different communication commands available in Unix/shell. And they are mail, news, wall & motd.
Q #50) How to find out the total disk space used by a specific user, say for example username is John?
Ans: The total disk space used by John can be found out as shown below.
du –s/home/John

Q #51) What is Shebang in a shell script?
Ans: Shebang is a # sign followed by an exclamation i.e. !. Generally, this can be seen at the beginning or top of the script/program. Usually, a developer uses this to avoid repetitive work. Shebang mainly determines the location of the engine which is to be used in order to execute the script.
Here '#' symbol is called as hash and '!' is called a bang.

Example: #!/bin/bash
The above line also tells which shell to use.

Q #52) What is the command to be used to display the shell's environment variables?
Ans: Command to be used to display the shell's environment variables is **env** or **printenv**.
Q #53) How to debug the problems encountered in shell script/program?
Ans: Though generally it depends on the type of problem encountered. Given below are some common methods used to debug the problems in the script.

- Debug statements can be inserted in the shell script to output/display the information which helps to identify the problem.
- Using "set -x" we can enable debugging in the script.

Q #54) How to know the variable length?
Ans: Variable length can be checked as shown below
$ {#variable}

Q #55) What is the difference between = and ==?
Ans: = -> This is used for assigning value to the variable.
== -> This is used for string comparison.

Q #56) How to open a read-only file in Unix/shell?
Ans: Read-only file can be opened as shown below:
vi –R <File Name>

Q #57) How can the contents of a file inside jar be read without extracting in a shell script?
Ans: The contents of the file inside a jar can be read without extracting in a shell script as shown below.
tar –tvf <File Name>.tar

Q #58) What is the difference between diff and cmp commands?
Ans: diff – Basically, it tells about the changes which need to be made to make files identical.
cmp – Basically it compares two files byte by byte and displays the very first mismatch.
Q #59) Explain in brief about sed command with an example.
Ans: sed stands for **stream editor**. And it is used for editing a file without using an editor. It is used to edit a given stream i.e. a file or input from a pipeline.
Syntax: sed options file
Example:
Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts_2017> echo "Hello World" | sed 's/Hello/Hi/'
```

Here '**s'** command present in **sed** will replace string **Hello** with **Hi**.
Output:

```
Hi World
```

Q #60) Explain in brief about awk command with an example.
Ans: awk is a data manipulation utility or command. Hence, it is used for data manipulation.
Syntax: awk options File Name
Example:
Script/Code

```
/u/user1/Shell_Scripts_2017> cat > script10
```

```
Hello John
Hello Richard
Hello Kevin
Hello Mike
Hello Robert
```

awk utility/command assigns variables like this.

$0 -> For whole line (e.g. Hello John)

$1 -> For the first field i.e. Hello

$2 -> For the second field

Execution over Shell Interpreter/Editor

```
awk '{print $0}' script10
```

The above script prints all the 5 lines completely.

Output:

```
Hello John
Hello Richard
Hello Kevin
Hello Mike
Hello Robert
```

Execution over Shell Interpreter/Editor

```
awk '{print $1}' script10
```

The above script prints only first word i.e. Hello from each line.

Output:

```
Hello
Hello
Hello
Hello
Hello
```