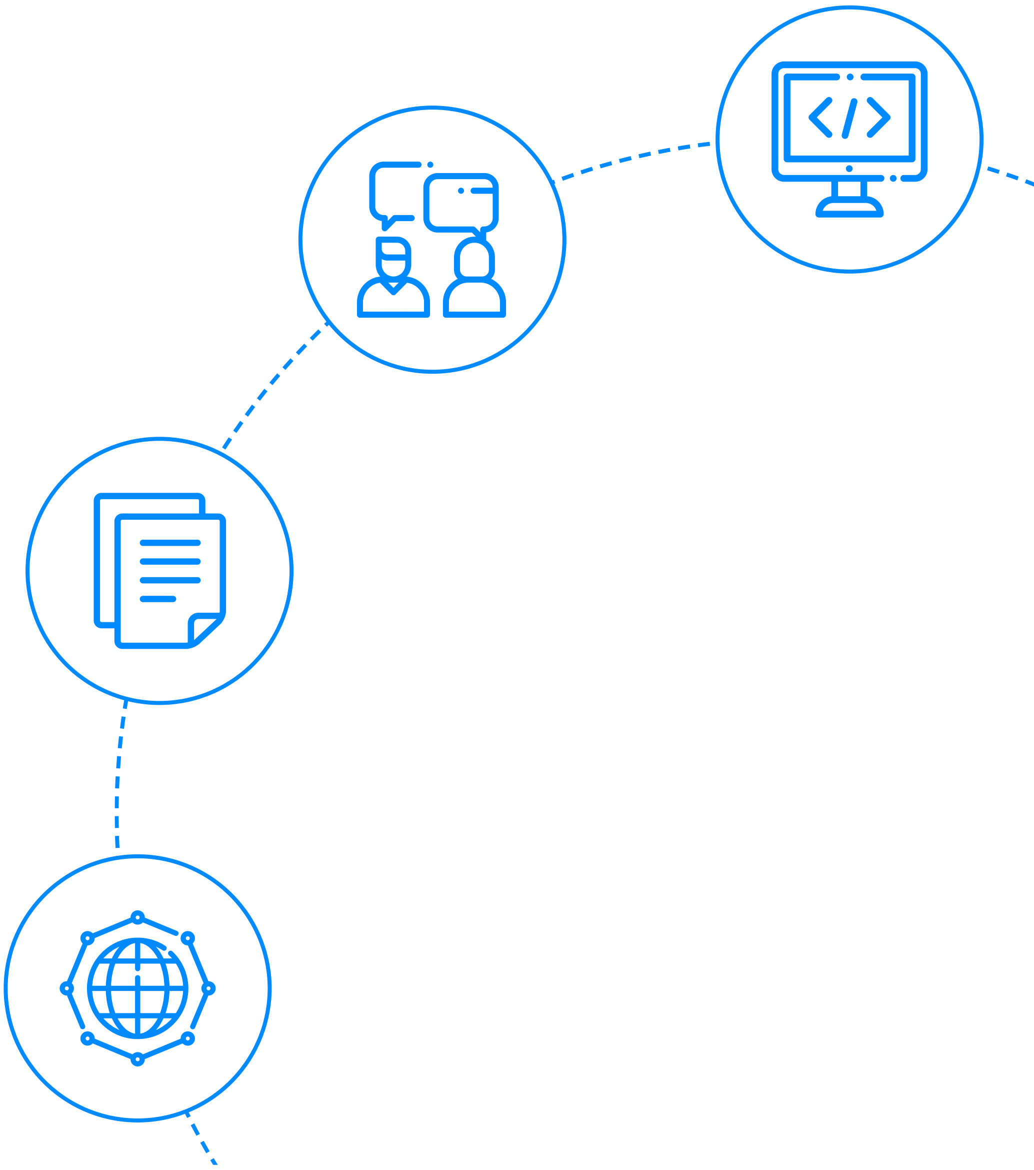
InterviewBit

Shell Scripting Interview Questions



Contents

Shell Scripting Interview Questions for Freshers

Shell Scripting Interview Questions for Freshers

(.....Continued)

21. What is the importance of $#?
22. Write the command that is used to execute a shell file.
23. Name the command that can be used to find out the number of arguments passed to the script?
24. How long does a variable in a shell script last?
25. What does the . (dot) indicate at the beginning of the file name? How can it be listed?
26. What is the use of the "$?" command?
27. What is the best way to run a script in the background?

Shell Scripting Interview Questions for Experienced

28. What do you mean by crontab?
29. Name two files of crontab command.
30. Name the command that is used to take the backup.
31. Write the difference between $* and $@
32. Name the command that is used to compare the strings in a shell script.
33. Write down the Syntax for all the loops in Shell Scripting.
34. What are interactive and non-interactive shells?
35. How will you pass and access arguments to a script in Linux?
36. What do you mean by the “s” permission bit in a file?
37. How will you debug a shell script?
38. What is the best way to debug the shell script/program problems?

Shell Scripting Interview Questions for Experienced

(.....Continued)

39. What is the importance of sed command and awk command?
40. Write the difference between “=” and “==”.
41. Name the command that is used to display the shell's environment variable.
42. Name different commands that are available to check the disk usage.
43. What do you mean by metacharacters?
44. How will you find total shells available in your system?
45. Explain how you will open a read-only file in Unix/Shell.
46. Name the command that one should use to know how long the system has been running.
47. Write the difference between $$ and $!?
48. Write difference between grep and find command.
49. How can we create a function in shell script?
50. How to use pipe commands?

Let's get Started

What is Shell Scripting

Text files containing commands to execute by a shell are called shell scripts. In this, long and repetitive series of commands are compiled into a single script that can be stored and executed at any time, thereby reducing programming efforts. In shell scripts, repetitive work is largely avoided. The most common reason for using shell scripting is to program operating systems of Windows, UNIX, Apple, etc. In addition, companies use this script to develop an operating system with their own features. In terms of system-level operations, it is considered to be the easiest programming language to use. A shell script is referred to as a batch file in DOS operating system, and EXEC in IBM's mainframe VM operating systems. Shell scripts can be used for the following applications:

- Automation of the code compiling process.
- Run a program or create an environment for programming.
- Complete batch processing and file manipulation.
- Integrate existing programs.
- Perform routine backups.
- Keeping an eye on a system.
- System administration's tasks
- Creating, maintaining, and implementing system boot scripts

A sample shell script:

```
echo "hello world"
```

Run it as follows:

```
$ bash hello.sh
# prints
hello world
```

Shell Scripting Interview Questions for Freshers

1. What is Shell?

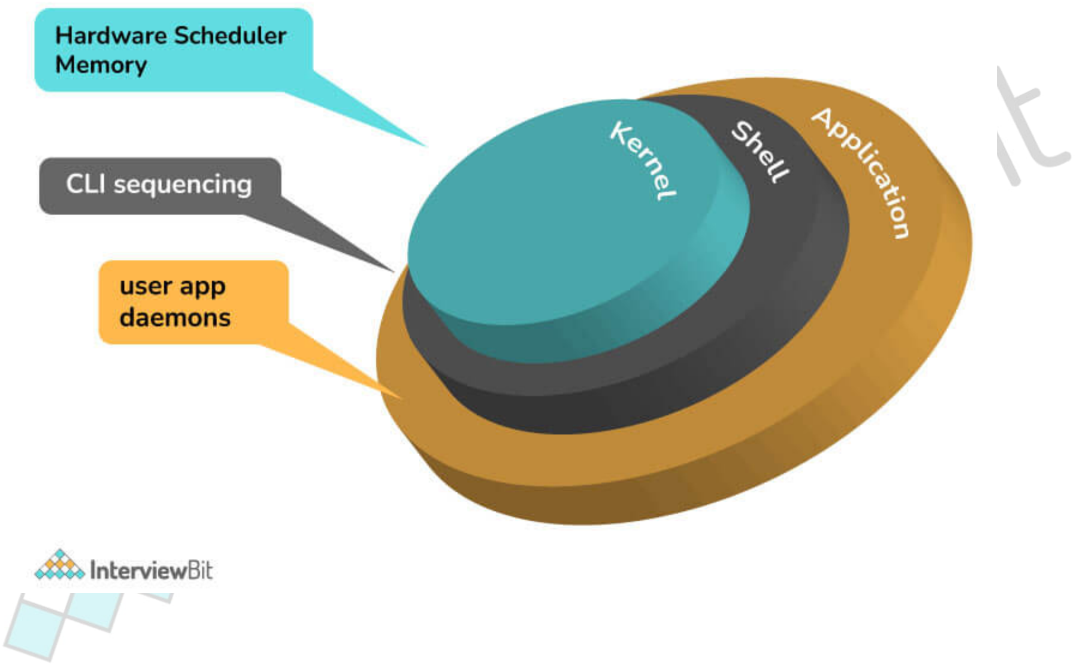
A Shell is basically a command-line interpreter between user and kernel or a complete environment specially designed to run commands, shell scripts, and programs. In this, whenever a user enters human-readable commands (input commands) through the keyboard, the shell communicates with the kernel to execute these commands, and display output in a shell script. Just as there are different flavors of operating systems, there are also different types of shells. Every shell has its own set of commands and functions. Shells issue the prompt, $, called a command prompt. You can type into the prompt while it is displayed.

After you press enter, your input is read by the shell. Based on the first word of your input, it determines the command you want to run. The characters in a word are separated using spaces and tabs.

Example:

Here is an example of a date command that displays the current date and time:

```
$date Tue Aug 10 06:03:35 MST 2021
```



2. Why is a shell script needed?

Shell scripts can be written for a variety of reasons:

- Keeping repetitive tasks to a minimum.
- Can be used by system administrators for routine backups.
- Monitoring the system.
- Adding new functions to the shell.
- Shell scripting allows you to create your own tools.
- System admin can automate daily tasks.

3. Write some advantages of shell scripting.

Shell scripting offers the following benefits:

- An interactive debugging tool, as well as a quick start.
- Programmers need not change their syntax since both command and syntax are identical to those entered directly into the command line.
- Shell scripts are easy to use and quicker to write.
- It helps automate administrative tasks, so it is time-saving.
- As shell scripts are written in an interpreted language, they can be run without any additional effort on almost any modern operating system, including UNIX, Linux, BSD, and Mac OS X.
- They can be utilized for bulk execution rather than single instructions.
- Using it, you can develop your own custom operating system with relevant features.
- Software applications can be developed according to their respective platforms with this tool.

4. Write some limitations of shell scripting.

Shell scripting has the following disadvantages:

- Errors are frequent and costly, and a single error can alter the command.
- The execution speed is slow.
- Bugs or inadequacies in the language's syntax or implementation.
- Large, complex tasks aren't well suited to it.
- Contrary to other scripting languages, etc., it provides a minimal data structure.
- Every time a shell command is executed, a new process is launched.

5. 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.

6. Name different types of shells available.

- Shells are divided into two categories:
    - **Bourne shell:** The $ character is the default prompt when using a Bourne-type shell.
    - **C shell:** The % character is the default prompt when using a C-type shell.

The Bourne-type is subdivided into the following categories:

- **Bourne Again shell (BASH):** This is the most common shell available on all Linux and based systems. It is open-source and freeware. In addition, it is an SH-compatible shell, with improved programming and interactive features over SH. It also allows you to efficiently perform many tasks.
- **Korn shell (KSH):** Korn is basically a Unix shell that was initially based on the Bash Shell Scripting. It's a high-level language that's quite advanced. It has associative arrays and handles the loop syntax better than Bash. It is basically an improved version of Bourne shell.

The C-type is subdivided into the following categories:

- **C shell (CSH):** C shell is almost like C itself since it uses the shell syntax of the C programming language. In most cases, a command is executed either interactively from a terminal keyboard or from a file.
- **TENEX/TOPS C shell (TCSH):** TCSH does not have a specific full name. TCSH is considered as an enhanced version of the CSH as it includes some additional features over CSH like command-line editing and filename or command completion. As with the previous version, it supports C-style syntax also.

7. Write difference between Bourne Shell and C Shell.

- **Bourne Shell:** It has compactness and speed that set it apart from other shells. But it lacks interactive features such as the ability to recall previous commands. Additionally, the Bourne shell does not support arithmetic and logical expressions.
- **C Shell:** It is a UNIX enhancement that incorporates interactive features like aliases and history of commands. Besides its built-in arithmetic and expression syntax, it also includes convenient programming features.

Difference between Bourne shell and C shell

- The C shell allows you to alias commands easily, whereas Bourne Shell does not allow this.
- In the C shell, long commands can be used repeatedly, but not in Bourne.
- Bourne does not have access to the command history, but the C shell does.
- In the case of C, there is no need to repeatedly type the command.

8. What do you mean by Shell variable?

Shell variables are integral parts of all Shell programs and scripts. In general, we know that variables usually store data either in the form of characters or numbers. Shell also stores and manipulates information using variables in its programs. Generally, shell variables are stored as strings. Variables in the shell provide the information needed for scripts/commands to execute. In the following example, a shell variable is created and then printed:

```
variable ="Hello"
echo $variable
```

9. What are different types of variables mostly used in shell scripting?

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.

10. Explain the term positional parameters.

In a shell program, positional parameters specify arguments that are used to launch the current process. A special set of variables is usually maintained by the shell for storing positional parameter values. Bash is an example of a shell that uses positional parameters. The bash shell can be used on Linux, BSD, macOS X, and Windows 10.

For example:

mycommand one five "six four”

In this case, the command name is mycommand, and there are four parameters in the command line: one, five, and "six four”.

Note: A space delimits each positional parameter and each thing after spaces are interpreted by the shell as individual parameters. Therefore, the parameter itself should be enclosed in quotation marks if it contains a space, as in "six four".

11. 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. The execution of a shell script proceeds in succession without these instructions. In shell programs, control instructions govern how execution flows.

12. How many types of control instructions are available in a shell?

Shells provide four types of control instructions as given below:

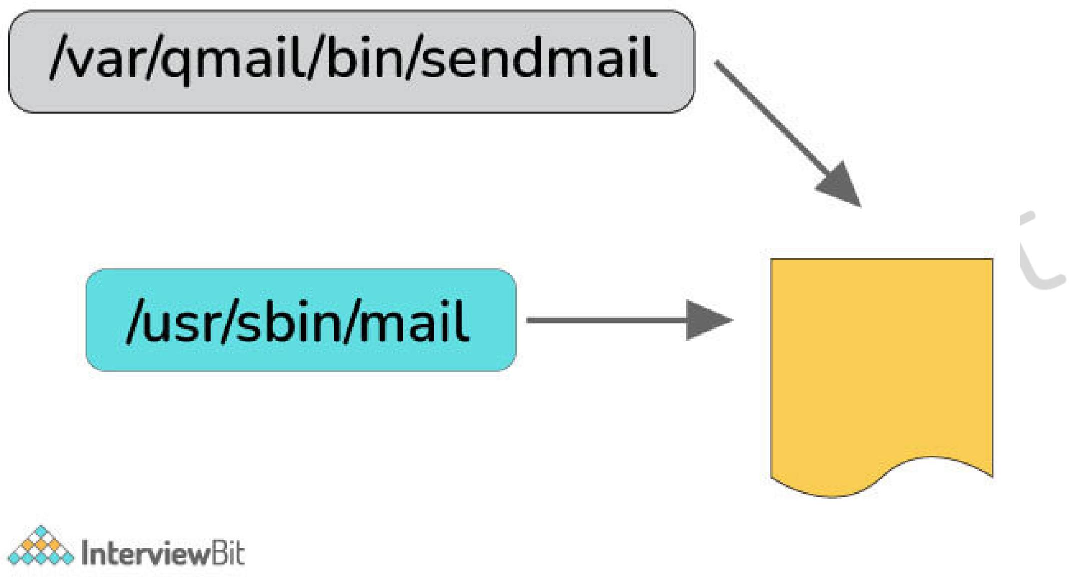
- **Sequence Control Instruction:** This assures that a program's command runs in the sequence they are listed. As soon as the first command in the sequence has been completed, the second command follows, and so on.
- **Decision Control Instruction:** It is also known as Selection Control Instruction. This instructs the computer which instruction it should execute next. In this, checking a condition is the basis for deciding which section to execute. If a given condition is True, a statement or set of statements will be executed; otherwise, they will be ignored.
- **Loop Control Instruction:** It is also known as Repetition Instruction. This instruction allows a computer to continuously run a particular sequence of code. In a loop structure, the statements can be repeated until a condition is True or False, a particular number of times, or once for each element within the collection
- **Case-Control Instruction:** This is used for selecting among several choices. Typically, they are used to execute only a particular block of statements within a series of statements.

13. Explain ways to create shortcuts in Linux.

Links present in Linux OS can be used to create shortcuts as given below:

- **Hard Link:** Hard links are mirror images of the originally linked files and are linked with an inode number. A hard-linked file remains even after the original file is deleted. Since hard links point to inodes, they cannot be implemented on directories. Hard links are created by the following command:

```
$ ln  [original filename] [link name]
```



- **Soft Link:** Generally, soft links (also referred to as Symbolic links) are linked to the file name and can reside on the same as well as different file systems. When a soft link is created or deleted, it does not affect the original file, but when the original file is deleted, the soft link stops working. Typically, soft links are aliases (alternative paths) for the original file. Soft links are created by the following command:

```
$ ln  -s [original filename] [link name]
```

14. How to check whether a link is a hard one or a soft link?

We can use -h and -L operators of the test command to check whether a link is hard or soft (symbolic link).

- **-h file   //true if the file is a symbolic link**
- **-L file   //true if the file is a symbolic link**

One can also use:

readlink FILE; echo $?   // This returns 1 if it's a hard link and 0 if it's a symbolic link.

15. Write difference between Hard link and Soft Link.

Hard Link	Soft Link
Hard-linked files have the same inode number.	Soft-linked files have different inode numbers.
Despite deleting the target file, it remains valid.	When an original file is deleted, it becomes invalid.
Hard links are faster than soft links.	Soft links are slower than hard links.
The command used for creating a hard link in Linux is "ln".	The command used for creating a soft link in Linux is “ln -s”.
There is no way to use it across file systems.	You can use it on any file system.
Hard links cannot use relative paths.	Both absolute and relative paths can be used in soft links.

16. What do you mean by GUI Scripting?

The GUI (Graphical User Interface) is a client/server interface that uses graphical icons and visual indicators to allow users to interact with devices. Specifically, it is used to control a computer and its applications. A wide variety of applications are supported, and it is largely dependent on the operating system.

17. What is the shebang line in shell scripting?

Shell scripts or programs often contain shebang at the top. In general, the Shebang (the series of characters "#!") is used to indicate the operating system to use a particular interpreter to process the rest of the file. Here, the symbol '#' is referred to as hash, and "!" is referred to as bang. This usually aids developers in avoiding repetitive work. Scripts are generally executed by the engine; therefore, shebang determines the location of the engine that will be used to run the script.

Example:

#!/bin/bash

In addition, this line also specifies which shell to use.

18. What is a file system? Write down the four core components of a Linux file system.

Generally, file systems are referred to as the collections of files, which include information related to those files. It would be impossible to tell where one piece of data stops and the next begins without a file system. There are four blocks in a file system:

- **Super Block:** A superblock contains information about a file system, including block size, group size, usage statistics, empty/filled blocks, inode table size & location, and so on. One of the tools used to describe a file system, along with inodes, entries, and files, is the superblock. Multiple superblocks are created with the file system, as the superblock contains critical information.
- **Boot Block:** Located on the disk label, a boot block is a special set of blocks that contains data or information on the disk layout. Normally, this block contains the bootstrap loader program, which a user runs upon launching the host computer. The boot block remains blank if the file system is not used for boot.
- **Data Block:** Also called storage blocks, data blocks contain the remainder of the space allocated to the file system. The data block's size is measured at the time of file system creation. For a regular file, the content of files is contained in the data blocks. For directories, the inode number and file name of the files are contained in the data blocks.
- **Inode:** Inodes contain information about each file in the filesystem. Normally, an inode doesn't contain a file's name, which is located in a directory instead. An inode contains information such as the type of file, the permission bits, the owner, the group, the file size, and the time when the file was modified.

19. Name the alternative command for echo.

The tput command is an alternative command for echo. We can use this command to control how the output is displayed. Additionally, shell scripts can do things such as underline text and center text, regardless of the size of the screen.

20. Explain the way you can connect to a database server.

Open client driver comes with the isql utility that can be used to connect to a database server from Linux. Here's how to do it:

isql –S serverName –U username –P password

21. 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.

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

Firstly, use the chmod command to set execute permission on your script as shown below:

chmod +x script-name-here.sh

Secondly, run or execute your script as follows:

./script-name-here.sh

Alternatively, you can execute shell script by:

sh script-name-here.sh

23. 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 $ #

24. 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.

25. 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. A Shell usually lists all the files except hidden files when you try to list the files in a shell. Despite this, hidden files can be found in the directory. The Is command must be run with the "–a" flag if you wish to see hidden files.

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

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

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

Example:

script.sh &

command &

Shell Scripting Interview Questions for Experienced

28. What do you mean by crontab?

Crontab stands for "cron table," meaning that it makes use of the cron scheduler to perform tasks. In other words, it is the list of commands that you wish to run on a regular schedule and the command that will let you manage it. It is possible to view or edit the table of commands using the crontab command. In addition to the schedule, the term "Crontab" also refers to the name of the program used to edit the schedule.

29. Name two files of crontab command.

cron.deny and **cron.allow** are two files in the /etc/cron.d directory that controls access to the crontab command. Crontab command tasks such as creating, editing, displaying, or removing crontab files are relegated to specific users through these files. Both files usually consist of a list of user names, one user name per line. Together, these access control files perform the following functions:

- cron.allow decides which users are allowed to run the crontab command.
- cron.deny decides which users are denied from using the crontab command.
- When cron.allow or cron.deny doesn't exist, superuser privileges are required to run it.

30. Name the command that is used to take the backup.

The backup is taken using the tar command. Tar is an acronym for tape archive and is used to create backups using tar, gzip, and bzip. Files can be saved and restored from and to a tape using this command. In this, files and directories are generally compressed into tarballs, an archive file. For this purpose, it is among the most widely used commands. Furthermore, the tarball can be easily moved from one server to another.

31. Write the difference between $* and $@

Unlike $@, where each quoted argument is treated as a separate argument, $* treats each positional parameter as a single argument.

32. Name the command that is used to compare the strings in a shell script.

To compare text strings, use the test command. By comparing each character in each string, the test command usually compares text strings. In most cases, tests are generally included as a conditional execution of shell commands. It can be used for:

- Comparison of file attributes
- Comparison of strings
- Comparing basic arithmetic operations

Using square brackets ([and]) around the EXPRESSION is equivalent to testing it with the test.

Syntax of test command:

- test EXPRESSION
- test EXPRESSION && true-command
- test EXPRESSION || false-command
- test EXPRESSION && true-command || false-command

Examples:

- test 50 -gt 40 && echo "Yes" || echo "No"

Because 50 is greater than 40, this command prints the text "Yes".

- ["Excellent" = "Excellent"]; echo $?

As the two strings are identical, this command prints "0" because the expression is true.

33. Write down the Syntax for all the loops in Shell Scripting.

In shell scripting, you can use three looping statements as given below:

- while statement
- for statement
- until statement

Syntax is as follows:

- **For Loop:** Loops that operate on lists of items are known as loops. Each item in the list receives the same set of commands.

Syntax:

```
 for var in word1 word2 ... wordN
do
        Statement to be executed
done
```

Example:

```
for a in 1 2 3 4 5 6 7 8 9 10
do
    if [ $a == 3 ]
    then
        break
    fi
    echo "Iteration no $a"
done
```

Output”

```
$bash -f main.sh
Iteration no 1
Iteration no 2
```

- **While Loop:** It involves evaluating a command first and then executing a loop based on the result. The loop will be terminated if the command raises to false.

Syntax:

```
while command
do
    Statement to be executed
done
```

Example:

```
a=0
while [ $a -lt 3 ]
do
    echo $a
    a=`expr $a + 1`
done
```

Output:

```
$bash -f main.sh
0
1
2
3
```

- **Until Loop:** As often as the condition/command evaluates to false, the until loop is executed. Once the condition or command becomes true, the loop terminates.

Syntax:

```
until command
do
         Statement to be executed until command is true
done
```

Example:

```
a=0
until [ $a -gt 3 ]
do
    echo $a
    a=`expr $a + 1`
done
```

Output:

```
$bash -f main.sh
0
1
2
3
```

34. What are interactive and non-interactive shells?

Interactive shell: /bin/bash and /bin/sh is interactive shell. It is a non-login shell that gets started from a command line. It first copies the parent environment and then invokes.

Non-Interactive shell: /sbin/nologin shell is non-interactive shell. It is present when the shell script is running and just inherits the parent's environment.

35. 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

36. What do you mean by the “s” permission bit in a file?

The SUID (Set User ID) bit is known as the "s" bit. SUIDS are special file permissions for executable files that enable other users to run the file as the file owner. There will be an s (to indicate SUID) special permission for the user instead of the normal x, which represents execute permissions. A file that has the "s" bit set will grant the process the rights of the file's owner while the program is running.

Example:

Executing the "passwd" command to change the password, for example, allows the user to write the new password to the shadow file even though its owner is "root".

37. How will you debug a shell script?

With set, you can turn on or off debugging options in the shell:

- **Set -x:** This displays commands and their arguments as they are being executed.
- **Set -v:** It displays shell input lines in real-time as they are read.

38. What is the best way to debug the shell script/program problems?

Following are some common methods of debugging a script:

- The shell script can be modified to include debug statements to display the information that can be useful in identifying the problem.
- By setting -x in the script, we can enable debugging.

39. What is the importance of sed command and awk command?

- **sed Command:** sed command is an acronym for stream editor. It is used for editing a file without using an editor. The SED command performs a variety of operations on files, such as searching, finding and replacing, inserting and deleting. However, substitution or find and replace are the most commonly used features of SED.
  Syntax: sed options file
  Example: Execution over Shell Interpreter/Editor

```
/u/user1/Shell_Scripts_2020> echo "Hi Gourav" | sed 's/Hi/Hello/'
```

In this case, “s” command in sed replaces the string Hi with “Hello”.

Output:

```
Hello Gourav
```

- **awk Command:** The command is a scripting language generally used to manipulate data and generate reports. There is no need to compile it and users are allowed to have access to a variable, string functions, numeric functions, and logical operations.
  Syntax: awk options File Name
  Example: Script/Code

```
/u/user1/Shell_Scripts_2017> cat > script10
Hello Aisha
Hello Rohan
Hello Gourav
```

An awk command/utility assigns variables in following way:

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

$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 3 lines completely.

Output:

```
Hello Aisha
Hello Rohan
Hello Gourav
```

Execution over Shell Interpreter/Editor

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

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

Output:

```
Hello
Hello
Hello
```

40. 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.
```

41. 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.

42. 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.

43. What do you mean by metacharacters?

In a data field or program, metacharacters are special characters that provide information about other characters. In shells, they're called regular expressions. A character that is neither a letter nor a number is generally considered a metacharacter. Using shell metacharacters, you can group together commands, redirect and pipe input/output, put commands in the background, reduce the size of file names and path names, etc. You can use them as wildcards to specify a file name without typing in the file's full name. The most common metacharacters are as follows:

- **Asterisk (*)**: Use the * to match any character.
- **Question Mark (?):** It matches a single character in the filename.
- **Brackets ([…]):** The metacharacters used here match some specified characters.
- **Hyphen (-):** When placed within [] (brackets), the - metacharacter matches a specified range.

44. 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
```

Execution over Shell Interpreter/Editor:

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

Output:

```
/bin/sh
```

/bin/bash

/sbin/nologin

/bin/ksh

/bin/csh

45. Explain how you will open a read-only file in Unix/Shell.

You can open a read-only file by: **vi –R <Filename>**

46. 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. You can also determine the number of users with running sessions, and the average system load for 1, 5, and 15 minutes. The information displayed at once can also be filtered based on your specified options.

Example: $ uptime

Entering the above command at the shell prompt, namely $ uptime, will produce the following output:

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

47. 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.

48. Write difference between grep and find command.

- **Grep command:** This command facilitates searching and displaying content based on regular expressions specified by the user. Using the Grep command, one can search in a file for patterns. Grep command have the following syntax: **grep "literal string" <filename>**

Example:

```
grep “apple” file1.txt                     //Displays all the lines with the word “appl
grep “apple” file1.txt  file2.txt   //Scan multiple documents and search the word “appl
```

- **Find command:** The FIND command searches for files and folders based on their size, modification time, and access time. You can use this command to search files and directories. Find command have the following syntax:
  find <path> <search criteria> <action>

Example:

```
find -type  f                             // Command will find all the files
find  -type  d                           //Command will find all the directories
find .  -name file1.txt        //Command will find file1.txt in the current directory.
```

49. How can we create a function in shell script?

In other programming languages, shell functions are much like subroutines, procedures, and functions. The syntax for declaring a function is as follows:

```
function_name ()
{
   list of commands
}
```

Function_name is the name of your function, and that's what you'll use to call it from anywhere in your script. The function name must be followed by parentheses, then a list of commands enclosed in braces.

Example:

The following example shows how function can be used:

```
#!/bin/sh
# Define your function here

Hello ()
  {
     echo "Hello World"
  }

# Invoke your function
Hello
```

The following output will be displayed after execution:

```
$./test.sh
Hello World
```

50. 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
```

Conclusion

You can automate a lot of repetitive manual tasks with shell scripting, which is an easy and powerful programming method. Modern programming languages offer many of the features that make them appealing to programmers. DevOps engineers and automation testers often find this concept vital in preparing for interviews. Shell scripting allows the development of simple to complex scripts. A number of daily tasks can be automated using shell scripting as well. We thought of making your job easier by putting together an ensemble of the most frequently asked interview questions and sample answers on shell scripting. These interview questions will definitely help you understand that shell is basically an interface between the user and the operating system, which interprets user commands to be executed by the kernel or operating system.

Useful Resources:

Unix Interview

Automation Testing Interview

Devops Interview

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions