**BASH/Linux Interview Question for DevOps Engineers**

[[Preetam Meena](https://okpreetam.medium.com/?source=post_page---byline--7e22928a1486---------------------------------------)](https://okpreetam.medium.com/?source=post_page---byline--7e22928a1486---------------------------------------)

[Preetam Meena](https://okpreetam.medium.com/?source=post_page---byline--7e22928a1486---------------------------------------)

·

Follow

7 min read

·

Jan 8, 2024

674

4

***1. What are the various types of variables used in shell scripting?***

There are 2 types of variables used in shell scripting they are as follows,

1. System-generated variables — As the name suggests these are the variables generated by the operating system. These variables are viewed by invoking the Set command.

2. User-defined variables — These are the variables that are created by the users, this can be viewed by invoking the echo command.

***2. How would you check a file with name example exist on a path or not***

The [ and test commands are essentially switches. They emit a true or false response, but considers both of them as successes. You can put this to use by pairing the commands with logical operators, such as && and ||. The && operator is executed when a response is true:

$ touch example  
$ test -e example && echo "foo"  
foo  
$ test -e notafile && echo "foo"  
$

The || operator executes when a response is false:

$ touch example  
$ test -e example || echo "foo"  
$ test -e notafile || echo "foo"  
foo  
$

If you prefer, you can use square brackets instead of test. In all cases, the results are the same:

$ touch example  
$ [ -e example &#93; && echo "foo"  
foo  
$ [ -e notafile &#93; && echo "foo"  
$

**[Tips for using the Linux test command](https://opensource.com/article/22/10/test-command-linux?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)**

[The [ and test commands are vital conditional statements when scripting.](https://opensource.com/article/22/10/test-command-linux?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)

[opensource.com](https://opensource.com/article/22/10/test-command-linux?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)

***3. What are the default permissions of the file when it is created?***

Any file created in Linux or UNIX has default permissions and to view the file permissions the unmask or user mask command is used for the newly created file. It is the 4-digit octal number used to set and express symbolic values. The default permission of the newly created files is**rw-rw-r** which is**664**.

***4. What is CRONTAB?***

It is used to represent the CRON Table. As it uses the CRON schedule to execute the tasks. It is the list of the commands used for executing the regular scheduler tasks. The scheduler is called CRONTAB.

***5. How to debug the problems encountered in the shell script and shell program?***

1. The first method is to put a debug command in the shell script program to output or display the error or bugs in the program
2. The second method is to use a set –x to debug the script.

***6. How would you know the disk usage using the shell script commands?***

There are 3 different methods to check the disk usage using the shell script commands they are as follows,

* dfspace command: — This is used in checking the free disk usage in terms of megabytes.
* df command: — This is used to check the free disk space
* du command: — This is used to check the directory-wise disk usage  
  ex: du -sh /data01/\*

***7. Give the purpose of the shebang line.***

The shebang line is at the top of each script used to determine the engine’s location and is used to execute the script.

***8. Tell about the four stages of the Linux Process.***

The four stages include:

* Waiting- The Linux process is waiting for the resource
* Running- The Linux process is currently running
* Stopped- The Linux process has been stopped after successful execution
* Zombie- The Linux process is still active in the process table, but it has stopped.

***9. What is a metacharacter?***

Metacharacter is a unique character included in a program or data field of a shell. It offers information about other characters. For example: to list all files beginning with character ‘s,’ use the ‘ls s\*’ command.

***10. Rename all .txt files to .log in a path***

ls | cut -d. -f1 | xargs -i mv {}.txt {}.log

***11. How to Read Command Line Arguments in Shell Scripts?***

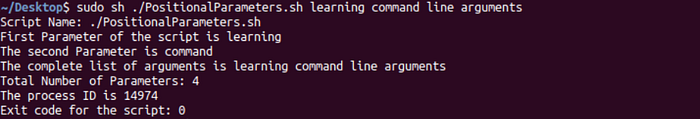
The bash shell has special variables reserved to point to the arguments that we pass through a shell script. Bash saves these variables numerically ($1, $2, $3, … $n)

We also have some special characters which are positional parameters, but their function is closely tied to our command-line arguments.

The special character **$#** stores the total number of arguments. We also have **$@** and **$\*** as wildcard characters which are used to denote all the arguments. We use **$$** to find the process ID of the current shell script, while **$?** can be used to print the exit code for our script.

#!/bin/sh  
echo "Script Name: $0"  
echo "First Parameter of the script is $1"  
echo "The second Parameter is $2"  
echo "The complete list of arguments is $@"  
echo "Total Number of Parameters: $#"  
echo "The process ID is $$"  
echo "Exit code for the script: $?"

./PositionalParameters.sh learning command line arguments



[How to Read Command Line Arguments in Shell Scripts? | DigitalOcean](https://www.digitalocean.com/community/tutorials/read-command-line-arguments-in-shell-scripts)

***12. Name standard streams in Linux.***

The standard streams in Linux are Standard Input, Standard Output, and Standard Error.

***13. Differentiate between $@ and $\****

$\* considers an entire set of positional arguments as a single string whereas, $@ treats each quote argument as a separate argument.

***14. how you will check if a file exists on the filesystem using the condition statement***

if [ -f /var/log/messages ]  
then  
echo "File exists."  
fi

***15. Use the echo command to get part of a string variable.***

echo ${variable:x:y}  
Where X is the starting position and y is the string length.

variable="Her name is Jen, and she is a developer."  
echo ${variable:12:6}   
  
#Output:  
#Jen, a  
#As J is at index 12 start from 0, and followed by 6 characters

***16. What is the diff between [[ $string == “efg\*” ]] and [[ $string == efg\* ]] ?***

[[ $string == efg\* ]] > checks if string begins with efg

[[ $string == “efg\*” ]] > checks if string is efg

*17.****SSH Authentication***

tail -n 10 /var/log/secure  
tail -n 10 /var/log/auth.log

SSH doesn’t like it if your home or *~/.ssh* directories have group write permissions. Your home directory should be writable only by you, ~/.ssh should be 700, and authorized\_keys should be 600 :

chmod 700 /home/user  
chmod 700 /home/user/.ssh  
chmod 600 /home/user/.ssh/authorized\_keys

**[SSH Authentication Refused: Bad Ownership or Modes for Directory](https://chemicloud.com/kb/article/ssh-authentication-refused-bad-ownership-or-modes-for-directory/?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)**

[Learn how to fix: "SSH Authentication Refused: Bad Ownership or Modes for Directory" error when trying to set up public…](https://chemicloud.com/kb/article/ssh-authentication-refused-bad-ownership-or-modes-for-directory/?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)

[chemicloud.com](https://chemicloud.com/kb/article/ssh-authentication-refused-bad-ownership-or-modes-for-directory/?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)

***18. What is the special shell variable $?***

The special shell variable $? in bash is used to get the exit status of the last command or the most recently executed process. You can use the $? variable to check if a command executed successfully or not, and take appropriate actions based on the result. A non-zero (1–255) exit status indicates failure.

#!/bin/bash  
  
# Try to copy a file that does not exist  
cp foo.txt bar.txt  
  
# Check the exit status of the cp command  
if [ $? -eq 0 ]; then  
 echo "Copy successful"  
else  
 echo "Copy failed"  
fi  
  
# Output:  
# cp: foo.txt: No such file or directory  
# Copy failed

**[How to return exit status codes in Bash](https://byby.dev/bash-exit-codes?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)**

[In Bash, exit codes (also known as return codes or status codes) are numeric values returned by executed commands or…](https://byby.dev/bash-exit-codes?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)

[byby.dev](https://byby.dev/bash-exit-codes?source=post_page-----7e22928a1486---------------------------------------" \t "_blank)

***19. What grep -oP '\d+' do?***

grep -oP '\d+' is used to search for and output only the numeric digits in the text. \d+ is a regular expression pattern that matches one or more digits. The -o flag ensures that only matching patterns are printed, and the -P flag enables Perl-compatible regular expressions.

echo "finnone-integration [Version=RevNo.201963]" | grep -oP '\d+'  
# Output: 201963

***20. how to list file names only in a list fashion.***

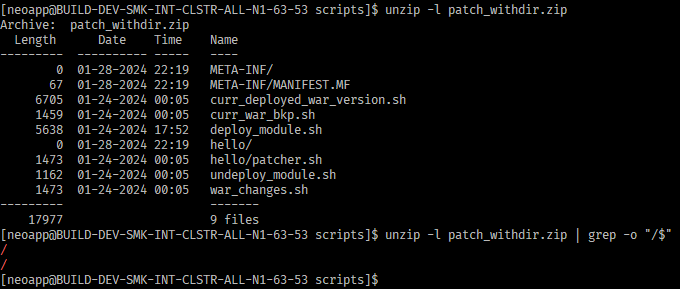
ls -1 will only show the filenames without any of the other attributes.

ls -1a will only show all filenames without any of the other attributes.

Store the output in a variable to use later filelist=`ls -1 /somedir/`

***21. how to find if a zip file containing a directory or not***

unzip -l yourfile.zip | grep -q "/$"



Explanation:

* The unzip -l command lists the contents of the zip file.
* The grep -q "/$" command searches for a line that ends with a forward slash, indicating a directory.
* The -q option is used to suppress the output, we are only interested in the return code.

If the zip file contains a directory, the command will exit with a return code 0. If the zip file does not contain a directory, the command will exit with a non-zero return code.

Example:

if unzip -l yourfile.zip | grep -q "/$"; then  
 echo "The zip file contains a directory."  
else  
 echo "The zip file does not contain a directory."  
fi

***22. When do we need curly braces around shell variables?***

To *read* from the variable (in other words, *expand* the variable), you must use $

$var # use the variable  
${var} # same as above  
${var}bar # expand var, and append "bar" too  
$varbar # same as ${varbar}, i.e expand a variable called varbar, if it exists.

Curly braces are also unconditionally required when:

* expanding array elements, as in ${array[42]}
* using parameter expansion operations, as in ${filename%.\*} (remove extension)
* expanding positional parameters beyond 9: "$8 $9 ${10} ${11}"