**Chapter 1**

Q1. What Is Linux?

Ans. Linux® is an [open source](https://www.redhat.com/en/topics/open-source/what-is-open-source) operating system (OS). An [operating system](https://www.redhat.com/en/technologies/linux-platforms/old-enterprise-linux) is the software that directly manages a system’s hardware and resources, like CPU, memory, and [storage](https://www.redhat.com/en/topics/data-storage/software-defined-storage). The OS sits between applications and hardware and makes the connections between all of your software and the physical resources that do the work.

Q2. What is Linux Kernel?

The Linux® kernel is the main component of a [Linux operating system (OS)](https://www.redhat.com/en/topics/linux/what-is-linux) and is the core interface between a computer’s hardware and its processes. It communicates between the 2, managing resources as efficiently as possible.

The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it’s a phone, laptop, server, or any other kind of computer.

The kernel has 4 jobs:

1. **Memory management:** Keep track of how much memory is used to store what, and where
2. **Process management:** Determine which processes can use the central processing unit (CPU), when, and for how long
3. **Device drivers:** Act as mediator/interpreter between the hardware and processes
4. **System calls and security:** Receive requests for service from the processes

Q3. What is Linux Shell?

The shell can be defined as a command interpreter within an operating system like Linux/GNU or Unix. It is a program that runs other programs. The shell facilitates every user of the computer as an interface to the Unix/GNU Linux system. Hence, the user can execute different tools/utilities or commands with a few input data.

Q4. Unix philosophy?

The Unix philosophy emphasizes building simple, compact, clear, modular, and [extensible](https://en.wikipedia.org/wiki/Extensibility) code that can be easily maintained and repurposed by developers other than its creators. The Unix philosophy favors [composability](https://en.wikipedia.org/wiki/Composability) as opposed to [monolithic design](https://en.wikipedia.org/wiki/Monolithic_application).

Q5. The role of shells in the Linux environment

Q6. Shebang

Q7. Setting up permissions on a script & Execute, Debug?

To check permissions - $ls -al

Allowing everyone

$ chmod +x <filename> or chmod 0755 <filename>

Only allow owner to execute the script :-

Chmod u+x <filename> or chmod 0700 <filename>

For permission for read and execute only :-

Chmod ug=rx <scriptname>

Remove read and execute permission for the group:-

Chmod ug= script.sh

<https://bash.cyberciti.biz/guide/Setting_up_permissions_on_a_script>

<https://www.cyberciti.biz/tips/debugging-shell-script.html>

Q8. Display the value of shell variables?

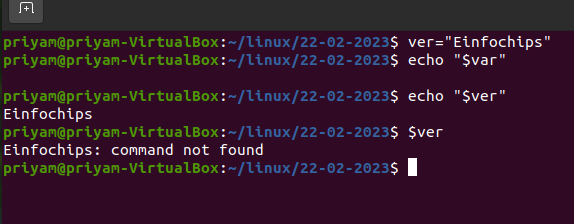
<https://www.digitalocean.com/community/tutorials/how-to-read-and-set-environmental-and-shell-variables-on-linux>

$printevn

$set

$set | less

$(set -o posix; set)



Q9. The export statement



Q10. Unset shell and environment variables

To unset

Text

Description automatically generated

Q11. Getting User Input Via Keyboard

<https://www.masteringunixshell.net/qa24/bash-how-to-read-from-keyboard.html#:~:text=To%20read%20input%20from%20the,variable%20use%20the%20read%20command.&text=If%20you%20don't%20give,while%20reading%20from%20a%20keyboard>.

Text

Description automatically generated

Q12. Perform arithmetic operations

Text

Description automatically generated

Q13. Create an integer variable

A picture containing text

Description automatically generated

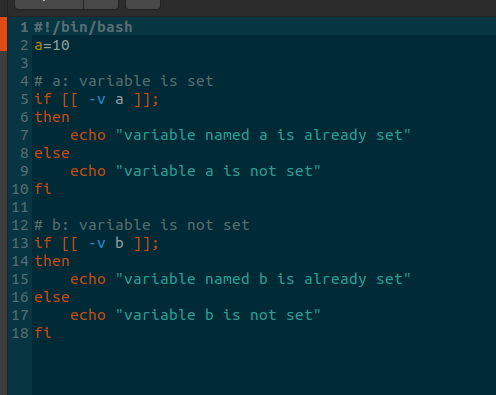
Q14. Create the constants variable

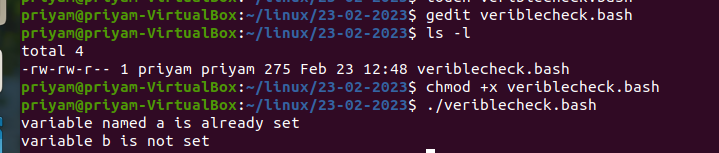


Text

Description automatically generated

Q15. Bash variable existence check





Q16. Recalling command history

Text

Description automatically generated

Q17. Path name expansion ?



Q18. Create and use aliases

Text

Description automatically generated

Q19. The tilde expansion Using aliases



Q20. Test command

**Chapter 2**

If structures to execute code based on a condition If..else..fi Nested ifs Multilevel if-then-else

Text

Description automatically generated

The exit status of a command?

$? Will be 0 1 and 2.

0 for true

1 for

Conditional execution

Numeric comparison?

Graphical user interface, text

Description automatically generated

String comparison

Text

Description automatically generated

Q. File attributes comparisons?

Text

Description automatically generated

Shell command line parameters

How to use positional parameters

Parameters Set by the Shell

Create usage messages

Exit command

The case statement

Dealing with case sensitive pattern.

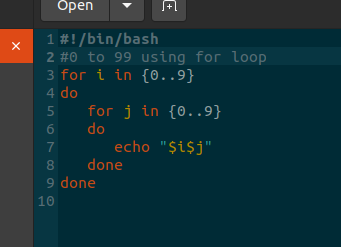
**Chapter 3**

The for loop statement

Text

Description automatically generated

Nested for loop statement



The while loop statement

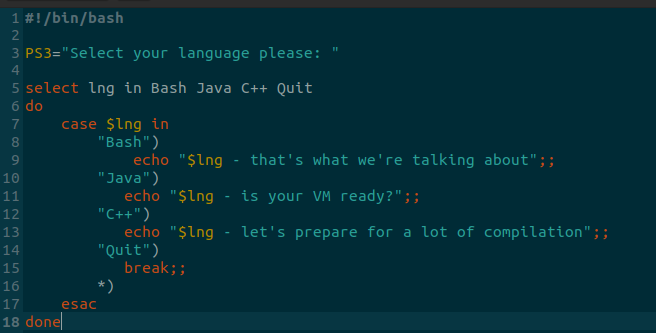
Use of : to set infinite while loop

The until loop statement

Text

Description automatically generated

The select loop statement? Exit the select loop statement? Using the break statement?



Using the continue statement

**Chapter 4**

Q1. Command substitution

Q2.Input and Output Standard input Standard output

Text

Description automatically generated

Q3. Standard error?

Q4. Empty file creation?

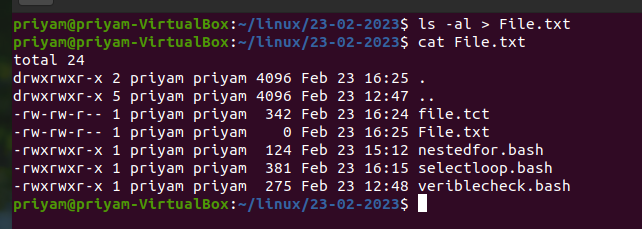
Touch command

Text

Description automatically generated

Q5. Redirection of standard error

Redirection of standard output



Appending redirected output

Redirection of both standard error and output

Writing output to files Assigns the file descriptor (fd) to file for output

Assigns the file descriptor (fd) to file for input

Closes the file descriptor (fd)

Opening the file descriptors for reading and writing

Reads from the file descriptor (fd)

Executes commands and send output to the file descriptor (fd)

**Chapter 5**

Q1. Linking Commands?

Q2. Multiple commands?

Q3. Putting jobs in background?

Q4. Pipes?

Q5. How to use pipes to connect programs?

Q6. Input redirection in pipes Output redirection in pipes?