**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

Q1 Which command is used to set terminal IO characteristic?

a) tty

b) ctty

c) ptty

**d) stty**

Q2 Which command is used to print a file ?

a) print

b) ptr

**c) lpr**

d) none of the above

Q3 What command do you use to create Linux file systems?

a) fdisk

**b) mkfs**

c) fsck

d) mount

Q4 What command is used to add printing jobs to the queue?

a) lpd

**b) lpr**

c) lpq

d) lpc

Q5 Which of the following command lists the last 10 lines of a file?

a) grep

**b) tail**

c) head

d) cat

Q6 User id 0 is

a) An innvalid user id

**b) The id of the root user \*\*\*\*\***

c) The id of a user when the user’s account is deleted

d) None of the mentioned

Q7 Which was the first Unix shell?

a) X Shell

**b) C Shell**

c) Bash Shell

d) Korn Shell

Q8 What service is used to translate domain names to IP addresses?

a) NFS

b) SMB

c) NIS

**d) DNS**

Q9 Which command runs the shell built-in command ‘command’ with the given argument?

**a) builtin**

b) caller

c) there is no command present for this purpose

d) none of the mentioned

Q10 The purpose of the PATH variable is to

a) Show the current directory

b) Show the directory path of a file

**c) Tells the shell what directories to search when a command is entered**

d) Both A and C

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

Q11 The permission -rwxr-xr-t represented in octal expression will be

a) 0777

**b) 1755**

c) 1754

d) 2754

Q12 After running this program, if you enter 1000, then what will be the output of the

program?

#!/bin/bash

echo "Please enter a number"

read a

if[ $a -lt 100 ]; then

echo "It is less than 100";

elif[ $a -lt 1000 ]; then

echo "It is less than 1000"

else

echo "It is greater than 1000"

fi

exit 0

**a) It is greater than 1000**

b) It is less then 1000

c) It is equal to 1000

d) None of the mentioned

Q13 What is the output of the following program?

[ -n $HOME ]

echo $?

[ -z $HOME ]

echo $?

**a) 0**

**1**

b) 1

0

c) 0

0

d) 1

1

Q14 What is the output of this program?

#!/bin/bash

echo "Which file do you want to check"

read x

until[ -e $x ]

do

echo "The file does not exist. Do you want to create? y/n"

read a

if[ $a = y ]; then

touch $x

echo "Your file has been created successfully."

fi

done

echo "The file is present in this directory"

exit 0

a) it checks the existance of your entered file in the present working directory

b) it creates the file if file does not exists

c) program runs untill you create the file

**d) all of the mentioned**

Q15 Indicate the right option to search for BOB, Bob, BOb or BoB?

a) grep –i Bob files

**b) grep ‘B[oO][bB]’ files**

c) grep ‘[BOB]’ files

d) grep -v ‘Bob’ files

**SECTION-C() (4x5 marks=20 marks)**

Q16 What are system calls? How these are implemented in Linux?

Ans-

A system call is a procedure that provides the interface between a process and the operating system. It is the way by which a computer program requests a service from the kernel of the operating system.

Different operating systems execute different system calls.

In Linux, making a system call involves transferring control from unprivileged user mode to privileged kernel mode; the details of this transfer vary from architecture to architecture. The libraries take care of collecting the system-call arguments and, if necessary, arranging those arguments in the special form necessary to make the system call.

System calls are divided into 5 categories mainly :

1.Process Control

2.File Management

3.Device Management

4.Information Maintenance

5.Communication

1. Process Control :

This system calls perform the task of process creation, process termination, etc.

The Linux System calls under this are fork() , exit() , exec().

fork()

A new process is created by the fork() system call.

A new process may be created with fork() without a new program being run-the new sub-process simply continues to execute exactly the same program that the first (parent) process was running.

It is one of the most widely used system calls under process management.

exit()

The exit() system call is used by a program to terminate its execution.

The operating system reclaims resources that were used by the process after the exit() system call.

exec()

A new program will start executing after a call to exec()

Running a new program does not require that a new process be created first: any process may call exec() at any time. The currently running program is immediately terminated, and the new program starts executing in the context of the existing process.

2.File Management :

File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc. The Linux System calls under this are open(), read(), write(), close().

open():

It is the system call to open a file.

This system call just opens the file, to perform operations such as read and write, we need to execute different system call to perform the operations.

read():

This system call opens the file in reading mode

We can not edit the files with this system call.

Multiple processes can execute the read() system call on the same file simultaneously.

write():

This system call opens the file in writing mode

We can edit the files with this system call.

Multiple processes can not execute the write() system call on the same file simultaneously.

close():

This system call closes the opened file.

3.Device Management :

Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc. The Linux System calls under this is ioctl().

ioctl():

ioctl() is referred to as Input and Output Control.

ioctl is a system call for device-specific input/output operations and other operations which cannot be expressed by regular system calls.

4.Information Maintenance:

It handles information and its transfer between the OS and the user program. In addition, OS keeps the information about all its processes and system calls are used to access this information. The System calls under this are getpid(), alarm(), sleep().

getpid():

getpid stands for Get the Process ID.

The getpid() function shall return the process ID of the calling process.

The getpid() function shall always be successful and no return value is reserved to indicate an error.

alarm():

This system call sets an alarm clock for the delivery of a signal that when it has to be reached.

It arranges for a signal to be delivered to the calling process.

sleep():

This System call suspends the execution of the currently running process for some interval of time

Meanwhile, during this interval, another process is given chance to execute

5.Communication :

These types of system calls are specially used for inter-process communications.

Two models are used for inter-process communication-

-Message Passing(processes exchange messages with one another)

-Shared memory(processes share memory region to communicate)

The system calls under this are pipe() , shmget() ,mmap().

pipe():

The pipe() system call is used to communicate between different Linux processes.

It is mainly used for inter-process communication.

The pipe() system function is used to open file descriptors.

shmget():

shmget stands for shared memory segment.

It is mainly used for Shared memory communication.

This system call is used to access the shared memory and access the messages in order to communicate with the process.

mmap():

This function call is used to map or unmap files or devices into memory.

The mmap() system call is responsible for mapping the content of the file to the virtual memory space of the process.

These are the various system calls involved in LINUX operating system.

Q17 Mention some ways to perform arithmetic operations in bash?

Ans-

Arithmetic operations can be done in multiple ways in bash. ‘let’, ‘expr’, ‘bc’ and double brackets are the most common ways to perform arithmetic operations in bash. The uses of these commands are shown in the following example.

Example:

#!/bin/bash

# Calculating the subtraction by using expr and parameter expansion

var1=$( expr 120 - 100 )

# print the result

echo $var1

# Calculate the addition by using let command

let var2=200+300

# Print the rsult

echo $var2

# Calculate and print the value of division using ‘bc’ to get the result

# with fractional value

echo "scale=2; 44/7" | bc

# Calculate the value of multiplication using double brackets

var3=$(( 5\*3 ))

# Print the result

echo $var3

Q18 How shell scripts are executed? Explain with example.

Ans-

Steps to write and execute a script

1.Open the terminal. Go to the directory where you want to create your script.

2.Create a file with .sh extension.

3.Write the script in the file using an editor.

4.Make the script executable with command chmod +x <fileName>.

5.Run the script using ./<fileName>.

Note: In the last step you have to mention the path of the script if your script is in other directory.

Example:

Hello World script-

Here we'll write a simple programme for Hello World.

First of all, create a simple script in any editor or with echo. Then we'll make it executable with chmod +x command. To find the script you have to type the script path for the shell.

# chmod +x hello\_world

./hello\_world

Look at the above, script echo Hello World is created with echo command as hello\_world. Now command chmod +x hello\_world is passed to make it executable. We have given the command ./hello\_world to mention the hello\_world path. And output is displayed.

Q19 How to use command-line arguments in bash?

Ans-

How to use command-line arguments in bash?

Command-line arguments are read by $1, $2, $3…$n variables. Command-line argument values are provided in the terminal when executing the bash script. $1 is used to read the first argument, $2 is used to read the second argument and so on.

Example:

#!/bin/bash

#Check any argument is provided or not

if [[ $# -eq 0 ]]; then

echo "No argument is given."

exit 0

fi

#Store the first argument value

color=$1

# Print the argument with other string

printf "You favorite color is %s\n" $color.