Operating System:

1. A program that acts as an intermediary between a user of a computer and the computer hardware
2. Defines an interface for the user to use services provided by the system
3. Creates an environment for the user

Unix: The Unix operating system is a set of programs that act as a link between the computer and the user.

Shell: The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

Kernel: The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.

Files and Directories: All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the filesystem.

Piping: A pipe is a form of redirection that is used in Linux to send the output of one program to another program for further processing.

Example: `cat sample | grep -v a | sort - r`

COMMANDS:

Command	Function
cat	concatenate and display reads data from the file and gives their content as output
echo	output a string
mkdir	make directories
cd	change directory
cp	copy files and directory
rm	remove files and directory
mv	move or rename files
rmdir	remove empty directories
tail	output the last part of files

Command	Function
file	determine file type
find	used to find files and directories and perform subsequent operations on them
who	get information about currently logged in user on to system. 1. Time of last system boot 2. Current run level of the system 3. List of logged in users and more.
passwd	used to change the user account passwords.
tty - short of teletype	prints the file name of the terminal connected to standard input allows you to interact with the system by passing on the data (you input) to the system, and displaying the output produced by the system
ls	list out all the files or directories available in a directory
whoami	list the account name associated with the current login
date +%d/%m/%Y	print out today's date
uname -nr	machine's name and version of the OS
clear	clear the screen
tput 10 30	moves the cursor to row 10, column 30
bc	used for command line calculator
tee	breaks the output of a program so that it can be both displayed and saved in a file does both the tasks simultaneously, copies the result into the specified files or variables and also display the result
script	makes a typescript of everything printed on your terminal
grep `[globally search a regular expression and print it]`	used to search for a string of characters in a specified file
touch	used to create, change and modify timestamps of a file
umask	used to set default permissions for files or directories the user creates
chmod `[changemode]`	used to change the access mode of a file

Command	Function
sort	used for printing lines of input text files and concatenation of all files in sorted order
cut	used for cutting out sections from each line of files and writing the result to standard output
pr	used to prepare a file for printing by adding suitable footers, headers, and the formatted text adds 5 lines of margin both at the top and bottom of the page
ps `[process status]`	used to list the currently running processes and their PIDs
kill	used to terminate processes manually sends a signal to a process which terminates the process
set	used to set or unset specific flags and settings inside the shell environment.

SHELL

Command	Function
read	takes the input from the keyboard and assigns it as the value of the variable
echo	echo command in linux is used to display line of text/string
expr	used to evaluate a given expression and display its standard output

SHELL OPERATORS

Command	Function
!	logical negation, this inverts a true condition into false and vice versa.
-o	logical OR, if one of the operands is true, then the condition becomes true.
-a	logical and, if both the operands are true, then the condition becomes true otherwise false.
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.

Command	Function
-gt	Checks if the value of the left operand if greater than the value of the right operand. if yes then condition becomes true.
-lt	Checks if the value of the left operand if less than the value of the right operand. if yes then condition becomes true.
-ge	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then the condition becomes true
-le	Checks if the value of left operand is less than or equal to the value of right operand, if yes then the condition becomes true

Operator	Description	Example
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a = $b] is not true.
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[$a != $b] is true.
-z	Checks if the given string operand size is zero; if it is zero length, then it returns true.	[-z $a] is not true.
-n	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.	[-n $a] is not false.
str	Checks if **str** is not the empty string; if it is empty, then it returns false.	[$a] is not false.

Question and Answers

1. Difference between terminal and shell?

Terminal	Shell
A terminal is a text input and output environment.	The shell is a command-line interpreter.
A terminal is a wrapper program that runs a shell and allows us to enter commands.	The shell is the program that actually processes commands and outputs results.
The terminal is a program that displays a graphical interface and allows you to interact with the shell.	A shell is a user interface for accessing the services of an operating system.

2. What is process status?

   Linux provides us a utility called ps which stands as abbreviation for “Process Status” and is used for viewing information related with the processes on a system

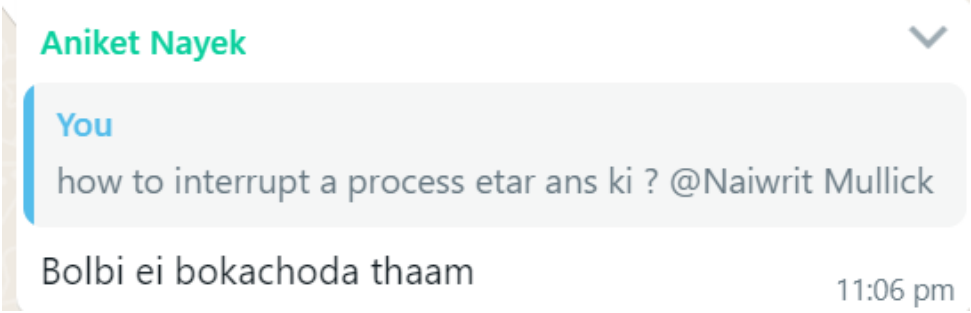
3. How can we find the running status?

   By using ps or top command

4. How to kill a process?

   Using kill command. To specify which process should receive the kill signal we need to provide the PID.

5. How to interrupt a process?



6. Functionality of top command?

   The top command displays the list of running processes in the order of decreasing CPU usage.

7. Shell Script and is requirement

   A shell script is a list of commands in a computer program that is run by the Unix shell which is a command line interpreter

   - Shell helps in doing work which is repetitive in nature. For example: When executing a bunch of commands, often, shells can take all these commands directly from a stored file and execute it, instead of writing them again every time.
   - We can transfer the shell script to other UNIX and similar operating systems and execute.

8. Multiply two numbers using shell script

   expr 10 * 5

9. Command to find all file accessed in the month of january

   ```
   ls -l | grep 'Jan'
   ```

10. How to change password in linux?

Step 1: To start, type password at the command prompt as shown below.

Step 2: Enter your old password, the one you're currently using.

Step 3: Type in your new password. Always keep your password complex enough so that nobody can guess it. But make sure, you remember it.

Step 4: You must verify the password by typing it again.

1. How can we convert file permission

   using chmod or umask

2. Difference between sh and bash
   - Bash is “Bourne Again SHell”, and is an improvement of the sh (original Bourne shell)
   - bash is sh, but with more features and better syntax
   - Shell scripting is scripting in any shell, whereas Bash scripting is scripting specifically for Bash
3. Difference between chmod and umask?

umask	chmod
used to set the default access permissions for files and directories which will be created in the future	used to change the access permissions for files that have been already created and are present in the system

4. How to set and calculate umask values for files and directories?
   1. We can use the umask command to set the default permissions with which the files/directories will be created.
   2. File -> The full permission set for a file is 666
      Directory -> The full permission set for a directory is 777
   3. the permissions will be calculated as (full permissions for directory) – (umask value)

5. Negative PID indicate?

   Negative PID values are used to indicate the process group ID.

6. Different types of Shell

C Shell	The Bourne Shell
The prompt for the shell is %	The prompt for this shell is $
? C shell also is known as csh ? Tops C shell also is known as tcsh	? POSIX shell also is known as sh ? Korn Shell also knew as ksh ? Bourne Again Shell also knew as bash (most popular)

7. Types of variable in Shell

   System variables - Created and maintained by Linux itself. This type of variable
   defined in CAPITAL LETTERS.

   User defined variables (UDV) - Created and maintained by user. This type of
   variable defined in lower letters.

8. Rules for naming variable

   1. Variable name must begin with Alphanumeric character or underscore character ( ) Variable name must begin with Alphanumeric character or underscore character (_), followed by one or more Alphanumeric character.
   2. Don't put spaces on either side of the equal sign when assigning value to variable.
   3. Variables are case-sensitive.
   4. You can define NULL variable
   5. Do not use ?,* etc, to name your variable names.

9. Command Substitution: the mechanism by which the shell performs a given set of commands and then substitutes their output in the place of the commands

10. Shell Loop Control
    - The break statement
    - The continue statement
    - The continue statement

Basic calculator in shell script

```
# !/bin/bash

# Take user Input
echo "Enter Two numbers : "
read a
read b

# Input type of operation
echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read ch
```

```
# Switch Case to perform
# calculator operations
case $ch in
1)res=`echo $a + $b | bc`
;;
2)res=`echo $a - $b | bc`
;;
3)res=`echo $a \* $b | bc`
;;
4)res=`echo "scale=2; $a / $b" | bc`
;;
esac
echo "Result : $res"
```