Name= shaik Raakin
Class =3rd sem
Branch = cybersecurity
usn=ENG24CY0192
Roll no:64
section:A
Assignment : 5

1Q) What is a shell in Linux OS? How many categories of shell is currently exists in Linux? Why bash shell is very popular in Linux distribution?
sol)
A shell is a command-line interface that enables users to collaborate with the Linux operating system by executing commands and scripts. Shells are divided into two primary types: command-line shells and graphical shells. There are several different shells commonly available, including bash (Bourne Again Shell), sh, csh, ksh, and zsh. bash is the most commonly used shell interface because it tends to be the most user friendly, supports the writing and execution of scripts, has a large set of features, and is the default shell interface for the vast majority of the same operating system distributions.

2Q). What does the ls -Z command display?
sol)
The ls -Z command is commonly used in Linux systems with SELinux (Security-Enhanced Linux) enabled. When invoked, the command will show the security context for each file and directory in addition to the standard ls output. A typical security context will consist of the SELinux user, role, type and level. The security context is important in controlling access and enforcing security policies outside the normal user/group/other traditional file permissions, making it easier for system administrators to manage and troubleshoot SELinux permissions.
Example:
raakin@DESKTOP-0FFMSPJ:~/uni_linx$ ls -Z
? Aarray.sh ? array.sh ? eq.sh ? eq1.sh ? hi.sh ? info.sh ? loc1.sh ? loc2.sh ? math.sh ? trial.sh
raakin@DESKTOP-0FFMSPJ:~/uni_linx$

3Q)Write a command to list all hidden files in the current directory.
sol)

In Linux, hidden files and directories start with a dot (.) and are not shown by default with normal listing commands. To view all hidden files in the current directory, the command is:

ls -a

The `-a` flag lists all files, including those starting with a dot. Alternatively, you can specifically list only hidden files using:

`ls -ld .?*`

The `-d` prevents listing directory contents recursively, while `.?*` matches all hidden files and directories.

Example:
raakin@DESKTOP-0FFMSPJ:~/uni_linx$ ls -a

. .. Aarray.sh array.sh eq.sh eq1.sh hi.sh info.sh loc1.sh loc2.sh math.sh trial.sh

4Q)Explain the difference between hard links and soft links (symbolic links) in Linux.

sol)

A hard link is a reference to the data on disk of a file. Multiple hard links to the same file reference the same data and share the same inode number. When the original file is deleted, the data is still accessible through the hard links. Hard links must exist on the same filesystem.

A soft link, also known as a symbolic link, is a type of file that contains a path to another file or directory. It behaves like a shortcut. Soft links can cross filesystem boundaries; hard links cannot. Soft links can link to directories whereas hard links cannot. If the file to which the soft link points is moved or deleted, the soft link file becomes invalid (broken). Soft links have their own inode.

5Q)A file has permissions -rwxr-x--x. Explain who can read, write, and execute it

sol)

In this permission string:

- The first character is a dash, indicating a file (rather than a directory).
- The next three characters (`rwx`) mean the owner (user) can read, write, and execute the file.
- The next three (`r-x`) mean the group can read and execute, but not write to the file.
- The last three (`--x`) mean others (everyone else) can only execute it, not read or write.

So, the owner has full control, the group can read and run the file, and others can only execute it, meaning they can run scripts or binaries but cannot see or modify the contents.

6Q)Write the command to change the group ownership of a file data.txt to group staff.

sol)

To change group ownership, use the chrgp command:

```
chgrp staff data.txt
```

This command sets the group owner of data.txt to staff. The user must be either the owner of the file or have root privileges, and the group “staff” should already exist on the system.

7Q)Why is it dangerous to give 777 permissions to a file? Explain with an example.
sol)
A file permission setting of 777 indicates a file can be accessed for reading, writing, and executing by anyone (owner, group, or others). This presents a number of security risks, including:

Anyone can read sensitive information or credentials.

An attacker can modify, corrupt, or delete the file.

If the file happens to be a script, a malicious user can then add malicious code to the script and execute it, compromising the entire system.

As an example, if you set permissions for a shell script containing important system commands to 777, anyone could modify those commands and possibly add commands that deleted critical system files or sent private information over the Internet.

8Q)What is the difference between apropos (i.e., man -k) and whatis (i.e., man -f)?
sol)
The command apropos (or man -k) queries the manual or documentation database for a keyword and will return a list of all commands and their descriptions which contain that keyword. It is helpful for finding related commands.

The command whatis (or man -f) will give a short description for a command, and pulls just the short description of the command.

So for example, if you use the command apropos copy it would return all commands with descriptions related to copying files. If you used the whatis command for cp whatis cp it would return one line description of what the cp command does.

9Q)Write a command to redirect the error output of a command to a file named error.log.

sol)

In Linux, file descriptor 2 represents standard error (stderr). To redirect only error output to error.log, use:

command_name 2> error.log

This means any errors produced by command_name will be saved to error.log instead of the terminal, making it easier to review errors.

10Q)0. How can you use the tee command to append output to a file instead of overwriting it?

sol)

The tee command allows you to read from the standard input and write to both the terminal and a file at the same time. By default, tee will overwrite the file, but if you'd rather append, use the -a option:

command | tee -a output.txt

Example:

```
raakin@DESKTOP-0FFMSPJ:~$ ping www.google.com | tee -a output.txt
PING www.google.com (142.250.67.228) 56(84) bytes of data.
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=1 ttl=118
time=73.1 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=2 ttl=118
time=30.6 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=3 ttl=118
time=41.9 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=4 ttl=118
time=31.2 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=5 ttl=118
time=65.3 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=6 ttl=118
time=29.0 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=7 ttl=118
time=96.8 ms
```

64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=8 ttl=118 time=60.7 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=9 ttl=118 time=48.3 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=10 ttl=118 time=30.2 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=11 ttl=118 time=30.9 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=12 ttl=118 time=59.9 ms
^Z
[1]+ Stopped   ping www.google.com | tee -a output.txt

Output:

raakin@DESKTOP-0FFMSPJ:~$ cat output.txt
PING www.google.com (142.250.67.228) 56(84) bytes of data.
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=1 ttl=118 time=73.1 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=2 ttl=118 time=30.6 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=3 ttl=118 time=41.9 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=4 ttl=118 time=31.2 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=5 ttl=118 time=65.3 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=6 ttl=118 time=29.0 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=7 ttl=118 time=96.8 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=8 ttl=118 time=60.7 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=9 ttl=118 time=48.3 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=10 ttl=118 time=30.2 ms
64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=11 ttl=118 time=30.9 ms

64 bytes from bom07s24-in-f4.1e100.net (142.250.67.228): icmp_seq=12 ttl=118 time=59.9 ms