**Assignment No. 1**

Name:- Kshitija Randive

Batch:- Devops 101

**---------Linux Interview Questions--------**

**Q1. What is the boot process in Linux?**

**Ans:-** The Linux boot process is the sequence of steps that start when a computer is turned on and ends when the system is ready for use. The process involves the BIOS, boot loader, kernel, init process, and user space.

Steps in the Linux boot process:

1. **BIOS or UEFI**

The BIOS or UEFI performs a Power-On Self-Test (POST) and checks the system's integrity.

2. **Boot loader**

The boot loader, such as GRUB or LILO, is loaded and finds the Linux kernel.

3. **Kernel**

The kernel initializes the system components and manages hardware interactions.

4. **Init process**

The init process, or systemd, takes over and launches system services.

5.**User space**

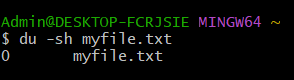
User space is initialized, allowing users to interact with the system through a login prompt or graphical interface.

**Q2. How can you create a zero-size file in Linux?**

With the help of touch command, we can create the zero-size file.

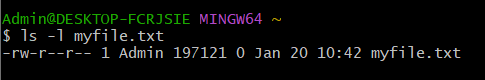


With the help of **du -sh myfile.txt** we can check the file size

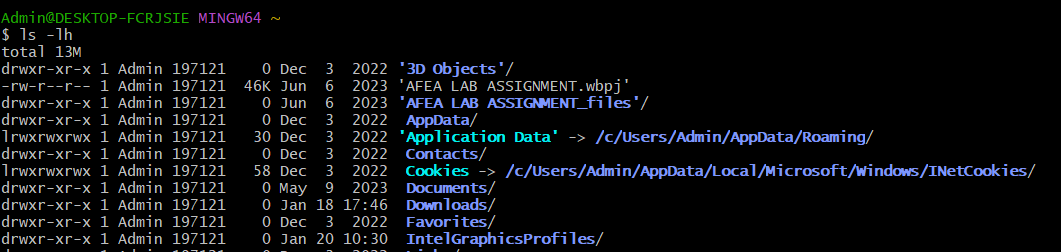


Also, with command we can check the rights as well the size in bytes

**ls -l myfile.txt**



Also, with **ls -lh** command you can see the size in KB, MB, GB or TB



**Q3. What are soft links and hard links in Linux? How do you create them? What are the differences between these two types of links?**

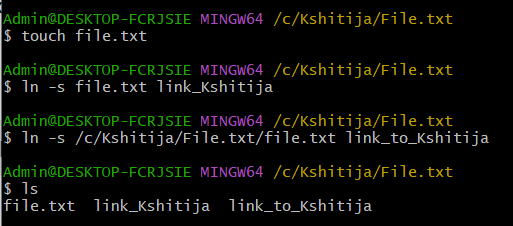
**Ans:- 1. Soft Links (Symbolic Links)**

A soft link (or symbolic link) is a special file that acts as a pointer to another file or directory. It stores the path of the target file rather than its actual data.

**Creating a Soft Link:**

Use the ln -s command:

ln -s target\_file link\_name



**Characteristics of Soft Links:**

* Acts as a shortcut to the target file.
* Can link to files or directories.
* If the original file is deleted, the soft link becomes **broken (dangling link)**.
* Soft links can span across different filesystems.

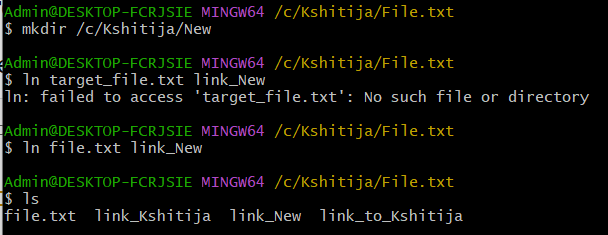
**2. Hard Links:**

A **hard link** is another name for an existing file. It points directly to the same inode (data block) as the original file.

**Creating a Hard Link:**

Use the ln command without -s:

ln target\_file link\_name



**Characteristics of Hard Links:**

* Both the original file and the hard link are identical; deleting one does not affect the other.
* Cannot be created for directories.
* Cannot span across different filesystems.
* Uses the same inode number as the original file.

**3. Differences Between Soft Links and Hard Links:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Soft Link (Symbolic Link)** | **Hard Link** |
| **Points to** | File path | Same inode (file data) |
| **File type** | Separate special file | Identical file |
| **Works across filesystems?** | Yes | No |
| **Works for directories?** | Yes | No |
| **Effect if the original file is deleted?** | Becomes broken (dangling) | Still accessible |
| **Does it use the same inode as the original?** | No | Yes |

**Q4. What is the first line typically written in a shell script? What is its meaning? What happens if this line is omitted, and how do you run the script in such a case?**

**Ans:-** The first line of a shell script is typically a shebang, which tells the operating system which program to use to execute the script. The shebang is a special comment that starts with the character #!

**Explanation:**

1. The shebang is a special comment that tells the operating system which program to use to execute the script.
2. The shebang is the first line of the file and starts with the character #!
3. The characters following the #! prefixes are interpreted as a path to an executable program.
4. For example, #!/bin/bash tells the operating system to use the /bin/bash program to interpret the script.
5. The script can then be executed by giving it execute permission.

**What Happens if the Shebang is Omitted?**

If you omit the #! line:

* The script runs using the default shell (which may not be Bash).
* On some systems, the default shell is sh, which may behave differently from bash.

**How to Run a Script Without a Shebang?**

If the shebang is missing, you can explicitly specify the shell when running the script:

* bash script.sh

or

* sh script.sh

This ensures that the script runs with the correct shell.

* The shebang (#!) defines the shell to use.
* Omitting it may lead to unpredictable behaviour.
* If omitted, explicitly run the script with a bash script.sh to ensure correct execution.

**Q5. How can you run a shell script in the background in Linux?**

**Ans:-**

* For simple background execution: ./script.sh &
* To keep running after logout: nohup ./script.sh &
* For interactive sessions: screen or tmux
* To detach an existing process: disown

**Q6. What is a crontab in Linux? Explain how it works and how to configure and schedule a job using crontab.**

**Ans:-** crontab (cron table) is a configuration file used by the **cron daemon** to schedule repetitive tasks in Linux. Cron automates the execution of scripts, commands, or programs at specified times or intervals.

**2. How Crontab Works**

**Cron daemon (**cron**)** runs in the background and executes scheduled tasks.

The **crontab file** contains a list of jobs along with their scheduled execution times.

Each user can have their own **crontab** file.

System-wide cron jobs are stored in /etc/crontab or /etc/cron.d/.

**3. Crontab Syntax and Format:** A crontab entry follows this format: MIN HOUR DOM MON DOW COMMAND

| Field | Allowed Values | Description |
| --- | --- | --- |
| MIN | 0 - 59 | Minute of execution |
| HOUR | 0 - 23 | Hour of execution |
| DOM | 1 - 31 | Day of the month |
| MON | 1 - 12 (or Jan-Dec) | Month |
| DOW | 0 - 7 (0 and 7 = Sunday) | Day of the week |
| COMMAND | Full path to script/command | Task to execute |

**Example:**

1. 4 \* \* 5 /home/user/script.sh

Runs /home/user/script.sh at **14:30 (2:30 PM) every Friday**.

**Q7. How do you allow ports in Linux?**

**And:-** There are 3 types to allow the port in Linux:

**For iptables:** Use the iptables -A INPUT -p tcp --dport PORT -j ACCEPT command.

Eg. ---sudo iptables -A INPUT -p tcp --dport 8080 -j ACCEPT

* -A INPUT: Appends the rule to the INPUT chain (for incoming traffic).
* -p tcp: Specifies the protocol (TCP in this case).
* --dport 8080: Specifies the destination port (8080 here).
* -j ACCEPT: Tells iptables to accept packets on this port.

**For firewalld:** Use the firewall-cmd --zone=public --add-port=PORT/tcp --permanent command.

Eg. --- sudo firewall-cmd --zone=public --add-port=8080/tcp --permanent

* --zone=public: Specifies the zone of the firewall (public is the default).
* --add-port=8080/tcp: Adds the rule to allow TCP traffic on port 8080.
* --permanent: Ensures that the rule is persistent after a reboot.

**For ufw:** Use the ufw allow PORT/tcp command.

Eg. ---sudo ufw allow 8080/tcp

**Multiple ports With** iptables

sudo iptables -A INPUT -p tcp -m multiport --sports 8080,9090,443 -j ACCEPT

**Multiple ports With** firewalld (multiple ports):

sudo firewall-cmd --zone=public --add-port=8080-8090/tcp --permanent

sudo firewall-cmd –reload

**Q8. How do you troubleshoot a remote server that is experiencing issues?**

9. What are the ping, telnet, curl, and wget

commands in Linux?

commands in Linux?

10. How can you check the status of services in a Linux machine?

11. How do you kill a process in Linux?

12. What are the nice and commands in Linux?

13. What is an inode in Linux?

14. How do you check CPU utilization in Linux?

15. What are the differences between the top and top commands?

16. What is a mount in Linux, and how do you create one?

17. How do you troubleshoot live logs in Linux?

18. What is the sed command in Linux?

19. What is the awk command in Linux?

20. What are the grep and egrep commands in Linux?

21. How can you list only directories in a Linux environment?

22. How do you check the processes running in Linux?

23. How do you get a Java thread dump in Linux?

24. How can you check the running ports on a Linux machine?

25. How do you declare a variable in a shell script?

26. What do $?, $#, and $\* represent in shell scripting?

27. How do you read a command line input in a shell script?

28. What is umask in Linux?

29. How do you change file permissions in Linux?

30. How can you connect to remote servers without a password? How is this achieved?

31. How do you open a file in read-only mode in the vi editor?

32. What is the purpose of the export command in Linux?

33. How do you send error logs and stdout logs to different files in Linux?

34. What is the nohup command in Linux?

35. What does the netstat command do in Linux?