**LINUX**

**3. Quick Reference Table**

| Action | Command |
| --- | --- |
| Create tar | tar -cvf archive.tar folder/ |
| Create tar.gz | tar -czvf archive.tar.gz folder/ |
| Create tar.bz2 | tar -cjvf archive.tar.bz2 folder/ |
| Create tar.xz | tar -cJvf archive.tar.xz folder/ |
| Extract tar | tar -xvf archive.tar |
| Extract tar.gz | tar -xzvf archive.tar.gz |
| Extract tar.bz2 | tar -xjvf archive.tar.bz2 |
| Extract tar.xz | tar -xJvf archive.tar.xz |

**1.what is FHS File hierarchy standards??**

The Filesystem Hierarchy Standard (FHS) is a standard that defines the directory structure and directory contents in Linux systems.

* It tells us what files should go where in the Linux filesystem.
* This helps system administrators, developers, and applications know where to store and find files consistently across Linux distributions.
* Most Linux distributions (Ubuntu, CentOS, Debian, Fedora) follow FHS.

| Directory | Purpose / Description |
| --- | --- |
| / | Root directory. The top-level directory. |
| /bin | Essential binary executables for all users (e.g., ls, cp). |
| /sbin | System binaries used by root/admin (e.g., ifconfig, iptables). |
| /usr | User programs and utilities. Non-essential system software. |
| /usr/bin | Binaries for normal users (applications, utilities). |
| /usr/sbin | System binaries not required for booting (admin tools). |
| /lib | Essential shared libraries required by binaries in /bin and /sbin. |
| /usr/lib | Libraries for /usr/bin and /usr/sbin. |
| /etc | Configuration files (system-wide). |
| /var | Variable data (logs, spool files, cache). |
| /tmp | Temporary files (cleared on reboot). |
| /home | Users’ home directories (personal files). |
| /boot | Boot loader files (kernel, initramfs). |
| /dev | Device files (hardware devices, virtual devices). |
| /proc | Virtual filesystem for kernel and process info. |
| /sys | Another virtual filesystem for kernel and device info. |
| /opt | Optional software packages or third-party apps. |
| /mnt or /media | Mount points for temporary or removable storage (USB, ISO). |

**2.what is linux boot process**

The Linux boot process is the sequence of steps a system follows from power-on to a fully operational state. It includes:

1. BIOS/UEFI stage: Performs hardware checks (POST) and locates a bootable device.
2. Bootloader stage: Loads the Linux kernel and initial RAM disk (initrd/initramfs). GRUB is commonly used.
3. Kernel stage: Initializes hardware, mounts the root filesystem, and starts the first process (init or systemd).
4. Init/Systemd stage: Starts essential system services and brings the system to the desired runlevel/target.
5. Runlevel/Target stage: Activates services according to the system’s mode (CLI, multi-user, or GUI).
6. Login stage: Presents a login prompt via terminal or display manager; the user shell or desktop environment starts, and the system is ready for use.

**3.what is soft link and hard link**

1. Hard Link

* A hard link is another name (alias) for the same inode of a file.
* Both the original file and hard link point to the same physical data on disk.
* Properties:
  1. Cannot link directories (except by root in some cases).
  2. Cannot span different filesystems.
  3. Deleting the original file does not delete the data as long as a hard link exists.
* Command to create a hard link:

ln original\_file hardlink\_file

2. Soft Link (Symbolic Link)

* A soft link is like a shortcut; it points to the path of the original file, not its inode.
* Properties:
  1. Can link directories.
  2. Can span different filesystems.
  3. If the original file is deleted, the soft link becomes broken.
* Command to create a soft link:

ln -s original\_file softlink\_file

**4.what is linux??what are the diff flavours of linux**

* Linux is an open-source, Unix-like operating system kernel developed by Linus Torvalds in 1991.
* It manages hardware resources (CPU, memory, storage) and provides a platform for applications.
* Linux is widely used in servers, desktops, embedded systems, and cloud environments.
* Key features: multitasking, multiuser, portability, security, stability.

2. Flavours (Distributions) of Linux

Linux comes in many distributions (distros), each with different package management, desktop environments, and target use cases. Some popular flavours:

| Flavour | Base / Package Manager | Typical Use Case |
| --- | --- | --- |
| Ubuntu | Debian-based (apt) | Desktop, servers, cloud |
| Debian | Debian (apt) | Servers, stability-focused |
| CentOS | RHEL-based (yum / dnf) | Enterprise servers, web hosting |
| Red Hat Enterprise Linux (RHEL) | RPM-based (yum / dnf) | Enterprise servers, paid support |
| Fedora | RPM-based (dnf) | Latest software, desktop & server |
| Arch Linux | Pacman | Minimalist, DIY setup, learning |
| openSUSE | RPM-based (zypper) | Servers, desktops |
| Kali Linux | Debian-based | Penetration testing, security tools |
| Linux Mint | Ubuntu-based | Desktop, user-friendly |

**5.what is shell?Different types shell**

**1. What is a Shell?**

* A shell is a command-line interpreter in Linux/Unix that allowsusers to interact with the operating system.
* It reads user commands, interprets them, and passes them to the kernel for execution.
* Think of it as a bridge between the user and the kernel.
* Shells can be interactive (accepting commands from the user) or scripted (executing commands from a file).

2. Different Types of Shells

Linux supports several types of shells. Some common ones:

| Shell Name | Abbreviation | Description / Features |
| --- | --- | --- |
| Bourne Shell | sh | Original shell, basic scripting support |
| Bourne Again Shell | bash | Most popular Linux shell, supports scripting, command history, tab completion |
| C Shell | csh | C-like syntax, used for scripting, limited features |
| TENEX C Shell | tcsh | Enhanced csh with command-line editing and history |
| Korn Shell | ksh | Combines features of sh and csh, good scripting capabilities |
| Z Shell | zsh | Advanced features, auto-completion, customization, widely used in modern setups |

**6.Difference between onpremise versus cloud.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  |  | | --- | --- | | **Onpremise** | **Cloud** | | Control of user is more | Control of user is less | | Infrastructure is not easy to scale | Infrastructure is easy to scale | | Internet connectivity is not need all the time | Internet is must for the services of the cloud | | Not available on subscription basis | Services available for purchase | | Cost is fixed | Cost is not fixed | | Secure is more | Secure is less | | Implementation is more | Implementation is less | | Data is easily portable | It is not easily portable | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**7.what is SSL and TL handshake ,why it is**

1. What is SSL / TLS?

* SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols used to secure data transmitted over a network.
* TLS is the modern, more secure version of SSL.
* Purpose:
  1. Encryption – Data is encrypted so that attackers cannot read it.
  2. Authentication – Ensures the server (and sometimes client) is legitimate using certificates.
  3. Integrity – Ensures data is not tampered with during transmission.

2. Why SSL/TLS is needed

* To protect sensitive information (passwords, credit cards, personal data) during transmission over the internet.
* To establish trust between client and server (e.g., HTTPS).

3. TLS/SSL Handshake

The handshake is the process by which the client and server establish a secure connection before exchanging data.

Steps of TLS Handshake:

1. Client Hello:
   * The client sends a message to the server with supported protocol versions, cipher suites, and a random number.
2. Server Hello:
   * The server responds with its chosen protocol version, cipher, and its digital certificate (proves its identity).
3. Certificate Verification:
   * The client verifies the server’s certificate using a trusted Certificate Authority (CA).
4. Key Exchange:
   * Both client and server agree on a shared session key (symmetric key) for encrypting the communication.
5. Finished Messages:
   * Both parties send a confirmation that the handshake is complete.
   * From now on, all communication is encrypted using the shared session key.

**8.what is TCP and UDP,why it is.**

Both TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are transport layer protocols in the OSI model, used for sending data over networks.

2. TCP (Transmission Control Protocol)

* Connection-oriented: Establishes a connection before data transfer.
* Reliable: Guarantees delivery of packets in correct order.
* Error checking: Retransmits lost packets.
* Use cases: Web browsing (HTTP/HTTPS), emails (SMTP/IMAP), file transfer (FTP).

3. UDP (User Datagram Protocol)

* Connectionless: Sends data without establishing a connection.
* Unreliable: No guarantee that packets arrive or are in order.
* Fast & lightweight: Minimal overhead.
* Use cases: Streaming (audio/video), online gaming, DNS queries.

4. Comparison Table

| Feature | TCP | UDP |
| --- | --- | --- |
| Connection | Connection-oriented | Connectionless |
| Reliability | Reliable, guarantees delivery | Unreliable, no guarantee |
| Ordering | Packets arrive in order | Packets may arrive out of order |
| Speed | Slower due to overhead | Faster, minimal overhead |
| Use Case | Web, email, FTP | Streaming, gaming, DNS |

**9.** **namespaces in Linux**

* A namespace in Linux is a feature that isolates system resources for a process or a group of processes.
* It allows processes to have their own separate view of system resources like process IDs, network interfaces, mounts, etc.
* Namespaces are the core technology behind containers (like Docker), providing process and resource isolation.

**2. Types of Namespaces**

Linux supports several types of namespaces:

| Namespace Type | Abbreviation | Purpose / Isolated Resource |
| --- | --- | --- |
| Mount | mnt | Isolates filesystem mount points |
| Process ID | pid | Each process sees its own set of PIDs |
| Network | net | Isolates network interfaces, IP addresses |
| UTS | uts | Isolates hostname and domain name |
| IPC | ipc | Isolates inter-process communication resources |
| User | user | Isolates user and group IDs |
| Cgroup | cgroup | Isolates control groups for resource limits |

**10.** **runlevel commands explain**

A runlevel is a mode of operation for a Linux system, defining which services and processes are started at boot.

Different runlevels are used for different system states:

| Runlevel | Purpose | Typical use |
| --- | --- | --- |
| 0 | Halt | Shuts down the system |
| 1 | Single-user mode | Maintenance, recovery |
| 2 | Multi-user mode without networking | Rarely used |
| 3 | Multi-user mode with networking (CLI) | Standard for servers |
| 4 | Undefined/custom | Usually unused |
| 5 | Multi-user with GUI | Standard for desktops |
| 6 | Reboot | Restarts the system |

**11.init levels implements in linux ?**