Batch: C-2 Roll No.: 16010122323
Experiment No. 01
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

TITLE: Exploring basic Commands of UNIX: Shell, Processes, Files

---

AIM: To Explore basic commands for handling File system under Unix/Linux using shell scripts.(Creating groups, chown , chmod , directory name, tty , diff, umask).

Expected Outcome of Experiment:

CO 1. To introduce basic concepts and functions of operating systems.

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate "Operating Systems", McGraw Hill Third Edition.**
3. **Sumitabha Das " UNIX Concepts & Applications", McGraw Hill Second**

Edition.

Pre Lab/ Prior Concepts:

An operating system (OS) is a resource manager. It takes the form of a set of software routines that allow users and application programs to access system resources (e.g. the CPU, memory, disks, modems, printers network cards etc.) in safe efficient and abstract way.

- The operating system kernel is in direct control of the underlying hardware. The kernel provides low-level device, memory and processor management functions (e.g. dealing with interrupts from hardware devices, sharing the processor among multiple programs, allocating memory for programs etc.)
- Basic hardware-independent kernel services are exposed to higher-level programs through a library of system calls (e.g. services to create a file, begin execution of a program, or open a logical network connection to another computer).
- Application programs (e.g. word processors, spreadsheets) and system utility programs (simple but useful application programs that come with the operating system, e.g. programs which find text inside a group of files) make use of system calls. Applications and system utilities are launched using a shell (a textual command line interface) or a graphical user interface that provides direct user interaction.

Operating systems can be distinguished from one another by the system calls, system utilities and user interface they provide, as well as by the resource scheduling policies implemented by the kernel.

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities.

Linux is a free open source UNIX OS for PCs.

Linux has all of the components of a typical OS :

- **Kernel**

The Linux kernel includes device driver support for a large number of PC hardware devices (graphics cards, network cards, hard disks etc.), advanced processor and memory management features, and support for many different types of file systems. In terms of the services that it provides to application programs and system utilities, the kernel implements most BSD and SYSV system calls, as well as the system calls described in the POSIX.1 specification.

The kernel (in raw binary form that is loaded directly into memory at system startup time) is typically found in the file /boot/vmlinuz, while the source files can usually be found in /usr/src/linux.

- **Shells and GUIs**

Linux supports two forms of command input: through textual command line shells similar to those found on most UNIX systems (e.g. sh - the Bourne shell, bash - the Bourne again shell and csh - the C shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers.

- **System Utilities**

Virtually every system utility that you would expect to find on standard implementations of UNIX has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more, and so on. These system utilities are designed to be powerful tools that do a single task extremely well (e.g. grep finds text inside files while wc counts the number of words, lines and bytes inside a file). Users can often solve problems by interconnecting these tools instead of writing a large monolithic application program.

- **Application programs**

Linux distributions typically come with several useful application programs as standard. Examples include the emacseditor, xv (an image viewer), gcc (a C compiler), g++ (a C++ compiler), xfig (a drawing package), latex (a powerful typesetting language) and soffice (StarOffice, which is an MS-Office style clone that can read and write Word, Excel and PowerPoint files).

Description of Commands and options:

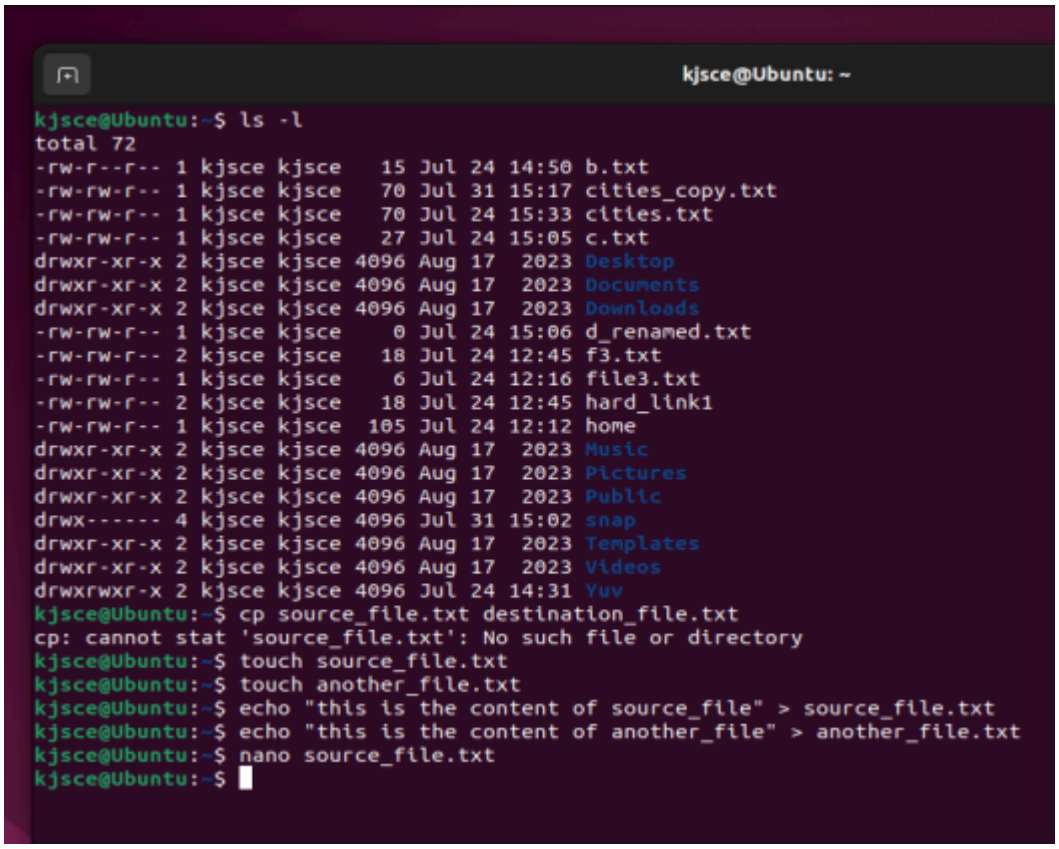
DOS commands: Attrib, dir, at, chkdsk, shutdown, tree, create a batch file, output and input redirection

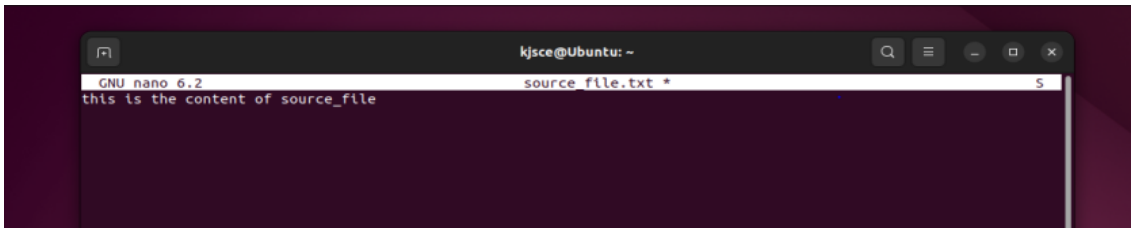
Windows utilities: msconfig, defragmenter, performance monitor, task manager, registry editor, event viewer, process explorer

Unix Commands:

1) Unix file operations: ls, cp, rm , mv, chmod, chown ,chgrp
2) Text file operations in Unix : cat , more , less , head, tail , grep
3) Unix directory management commands : cd, pwd , ln, mkdir, rmdir
4) Unix system status commands: hostname, w, uname
5) Process management: ps, top, kill
6) Unix users commands: whoami , id, groups, passwd , who, last

Implementation details:





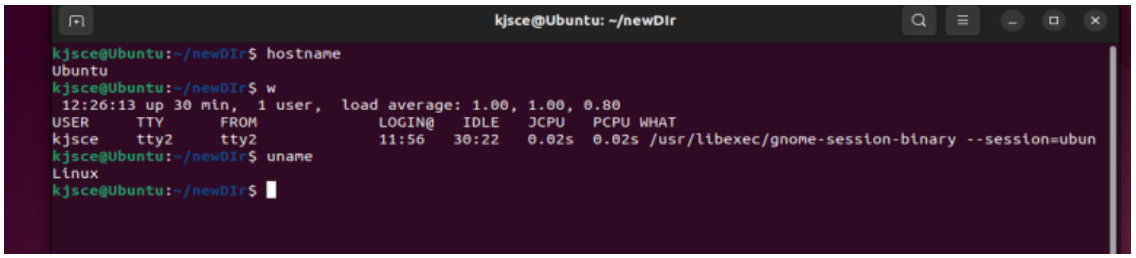
```
kjsce@Ubuntu:~$ nano source_file.txt
kjsce@Ubuntu:~$ touch file1.txt
kjsce@Ubuntu:~$ touch file2.txt
kjsce@Ubuntu:~$ cp file1.txt file2.txt
kjsce@Ubuntu:~$
```

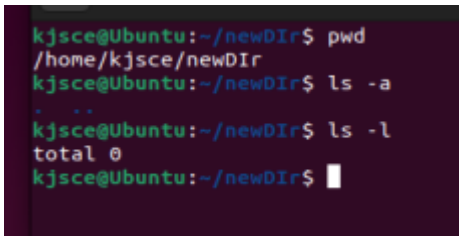
```
kjsce@Ubuntu:~$ touch oldname.txt
kjsce@Ubuntu:~$ mv oldname.txt newname.txt
kjsce@Ubuntu:~$
```

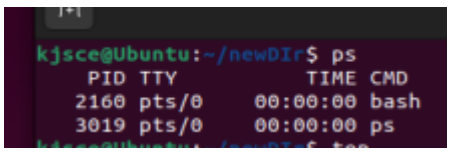
```
kjsce@Ubuntu:~$ touch script.sh
kjsce@Ubuntu:~$ nano script.sh
kjsce@Ubuntu:~$ chmodn 755 script.sh
Command 'chmodn' not found, did you mean:
  command 'chmod' from deb coreutils (8.32-4.1ubuntu1.2)
Try: apt install <deb name>
kjsce@Ubuntu:~$ chmod 755 script.sh
kjsce@Ubuntu:~$ ls -l script.sh
-rwxr-xr-x 1 kjsce kjsce 22 Aug  1 12:18 script.sh
kjsce@Ubuntu:~$
```

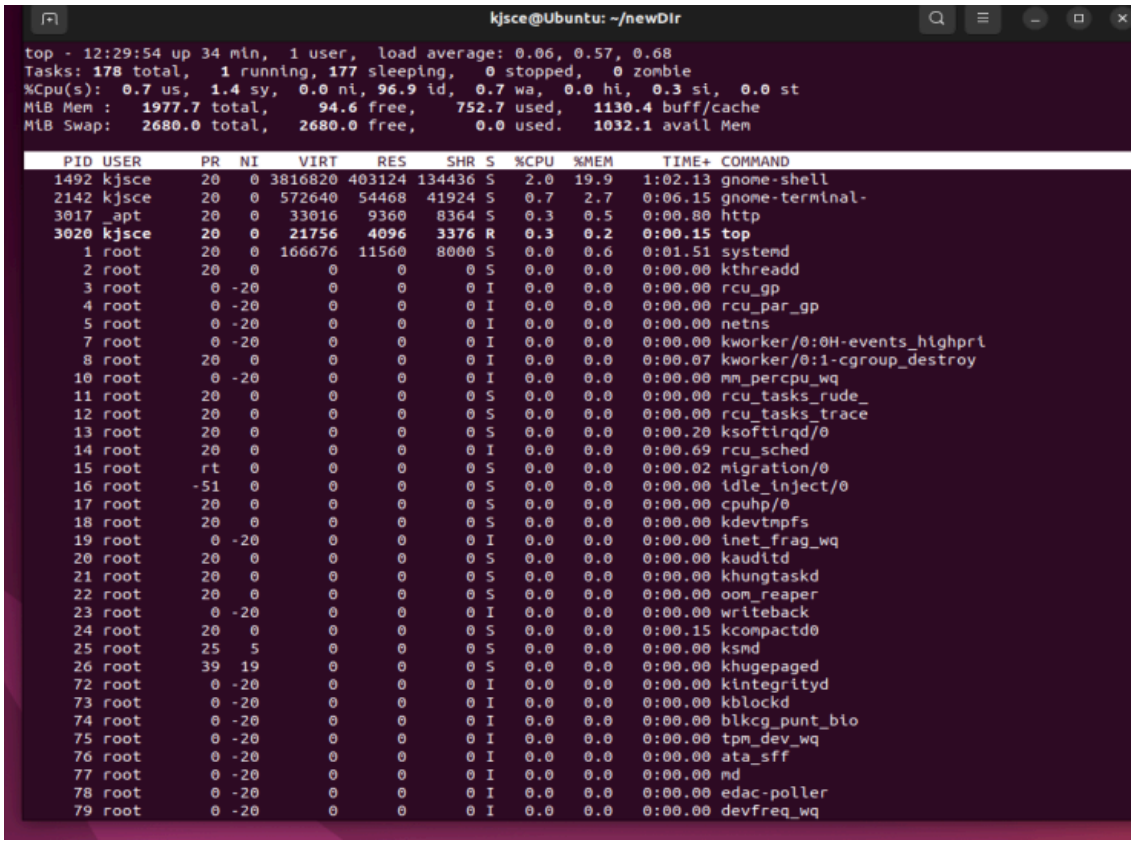
```
kjsce@Ubuntu:~$ cat file.txt
kjsce@Ubuntu:~$ touch file1.txt
kjsce@Ubuntu:~$ touch file2.txt
kjsce@Ubuntu:~$ cat file1.txt file2.txt > combined.txt
kjsce@Ubuntu:~$ ls combined.txt
combined.txt
kjsce@Ubuntu:~$
```

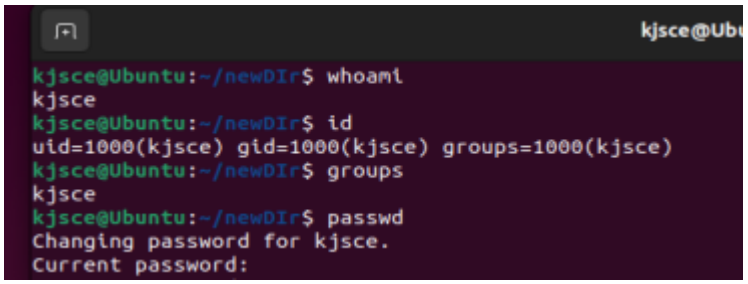
```
kjsce@Ubuntu:~$ mkdir newDIr
kjsce@Ubuntu:~$ cd newDIr/
kjsce@Ubuntu:~/newDIr$ cd..
cd..: command not found
kjsce@Ubuntu:~/newDIr$ cd ..
kjsce@Ubuntu:~$ cd newDIr/
kjsce@Ubuntu:~/newDIr$
```

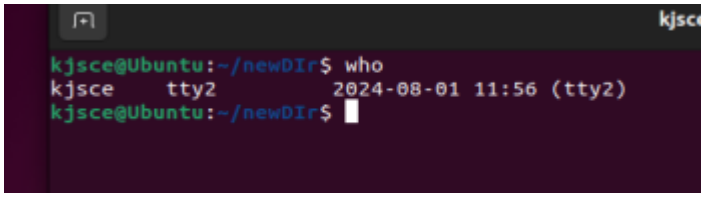

kjsce@Ubuntu: ~/newDIr
kjsce@Ubuntu:~/newDIr$ hostname
Ubuntu
kjsce@Ubuntu:~/newDIr$ w
12:26:13 up 30 min, 1 user, load average: 1.00, 1.00, 0.80
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
kjsce tty2 tty2 11:56 30:22 0.02s 0.02s /usr/libexec/gnome-session-binary --session=ubun
kjsce@Ubuntu:~/newDIr$ uname
Linux
kjsce@Ubuntu:~/newDIr$


kjsce@Ubuntu:~/newDIr$ pwd
/home/kjsce/newDIr
kjsce@Ubuntu:~/newDIr$ ls -a
. ..
kjsce@Ubuntu:~/newDIr$ ls -l
total 0
kjsce@Ubuntu:~/newDIr$


kjsce@Ubuntu:~/newDIr$ ps
PID TTY TIME CMD
2160 pts/0 00:00:00 bash
3019 pts/0 00:00:00 ps


kjsce@Ubuntu: ~/newDir
top - 12:29:54 up 34 min, 1 user, load average: 0.06, 0.57, 0.68
Tasks: 178 total, 1 running, 177 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 1.4 sy, 0.0 ni, 96.9 id, 0.7 wa, 0.0 hi, 0.3 si, 0.0 st
MiB Mem : 1977.7 total, 94.6 free, 752.7 used, 1130.4 buff/cache
MiB Swap: 2680.0 total, 2680.0 free, 0.0 used. 1032.1 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1492 kjsce 20 0 3816820 403124 134436 S 2.0 19.9 1:02.13 gnome-shell
2142 kjsce 20 0 572640 54468 41924 S 0.7 2.7 0:06.15 gnome-terminal-
3017 _apt 20 0 33016 9360 8364 S 0.3 0.5 0:00.80 http
3020 kjsce 20 0 21756 4096 3376 R 0.3 0.2 0:00.15 top
1 root 20 0 166676 11560 8000 S 0.0 0.6 0:01.51 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns
7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
8 root 20 0 0 0 0 I 0.0 0.0 0:00.07 kworker/0:1-cgroup_destroy
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
11 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_tasks_rude_
12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_tasks_trace
13 root 20 0 0 0 0 S 0.0 0.0 0:00.20 ksoftirqd/0
14 root 20 0 0 0 0 I 0.0 0.0 0:00.69 rcu_sched
15 root rt 0 0 0 0 S 0.0 0.0 0:00.02 migration/0
16 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
17 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
18 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
19 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 inet_frag_wq
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kauditd
21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd
22 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oom_reaper
23 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 writeback
24 root 20 0 0 0 0 S 0.0 0.0 0:00.15 kcompactd0
25 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksmd
26 root 39 19 0 0 0 S 0.0 0.0 0:00.00 khugepaged
72 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kintegrityd
73 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kblockd
74 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 blkcg_punt_bio
75 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 tpm_dev_wq
76 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ata_sff
77 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 md
78 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 edac-poller
79 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 devfreq_wq


kjsce@Ubu
kjsce@Ubuntu:~/newDir$ whoami
kjsce
kjsce@Ubuntu:~/newDir$ id
uid=1000(kjsce) gid=1000(kjsce) groups=1000(kjsce)
kjsce@Ubuntu:~/newDir$ groups
kjsce
kjsce@Ubuntu:~/newDir$ passwd
Changing password for kjsce.
Current password:


kjsc
kjsce@Ubuntu:~/newDir$ who
kjsce tty2 2024-08-01 11:56 (tty2)
kjsce@Ubuntu:~/newDir$

Conclusion:

In this experiment, we have explored fundamental Unix commands related to process management and user operations.

Post Lab Descriptive Questions

1. Explain how do you read and interpret syntax of any OS command.

To read and interpret the syntax of any OS command, follow these steps:

1. **Command Name:** Identify the command being used. This is typically the first word in the syntax.
2. **Options/Flags:** Options or flags modify the behavior of the command. They usually start with a hyphen (-) or double hyphen (--).
3. **Arguments:** These are the inputs to the command. They can be files, directories, or other data that the command will act upon.

Syntax Structure: Understand the structure of the command which usually follows a pattern such as:
css
Copy code

```
command [options] [arguments]
```

4.

Man Pages: Use the manual pages (man command) to get detailed information about the command syntax, options, and examples.
bash
Copy code

```
man <command>
```

5.

For example, the `ls` command in Unix/Linux:

bash

Copy code

```
ls -l /home/user
```

- `ls` is the command.
- `-l` is an option to display detailed information.
- `/home/user` is the argument specifying the directory to list.

2. Explain different functions of the operating systems.

The operating system (OS) performs several critical functions, including:

1. **Process Management:** Handles the creation, scheduling, and termination of processes. Ensures efficient CPU usage.
2. **Memory Management:** Manages the allocation and deallocation of memory space to various programs to optimize system performance.
3. **File System Management:** Manages files on disk, including file creation, deletion, reading, writing, and organization within directories.
4. **Device Management:** Manages device communication via drivers, allowing the OS to interface with hardware components like printers, disks, and keyboards.
5. **Security and Access Control:** Ensures that unauthorized users do not access the system and controls permissions for accessing resources.
6. **User Interface:** Provides a user interface, either command-line (CLI) or graphical (GUI), for user interaction with the system.
7. **Networking:** Manages network connections, enabling communication between computers and sharing of resources.
8. **System Performance Monitoring:** Monitors system performance and provides tools to manage and optimize resource usage.

3. What are the default permissions assigned by Unix for Directory.

In Unix, the default permissions assigned to a newly created directory are typically `755`. This can be broken down as follows:

- **Owner (User):** Read (r), write (w), and execute (x) permissions.

- **Group:** Read (r) and execute (x) permissions.
- **Others:** Read (r) and execute (x) permissions.

In symbolic notation: `rwxr-xr-x` In octal notation: `755`

These default permissions can be modified using the `umask` command which sets the default file creation permissions for new files and directories.

4. Give the difference between DOS and WINDOWS.

DOS (Disk Operating System):

1. **Interface:** Command-line interface (CLI).
2. **Multitasking:** Single-tasking (can run one program at a time).
3. **File System:** Uses FAT12, FAT16, or FAT32 file systems.
4. **Networking:** Limited networking capabilities.
5. **User Interaction:** Requires users to memorize and type commands.
6. **Applications:** Runs DOS-based applications.

Windows:

1. **Interface:** Graphical user interface (GUI).
2. **Multitasking:** Supports multitasking (can run multiple programs simultaneously).
3. **File System:** Uses NTFS (New Technology File System), FAT32.
4. **Networking:** Advanced networking capabilities, including internet and intranet connectivity.
5. **User Interaction:** User-friendly with point-and-click navigation.
6. **Applications:** Runs a wide range of applications, including modern software with rich graphical interfaces.

5. Explain Booting Process.

The booting process is the sequence of steps that the computer performs when it is powered on to load the operating system and prepare it for use. The process typically includes:

1. **Power-On Self Test (POST):** The system's firmware (BIOS/UEFI) performs initial hardware checks to ensure all components are functioning properly.
2. **Boot Loader Execution:** The BIOS/UEFI locates the boot loader on the bootable device (hard drive, SSD, USB, etc.) and loads it into memory.
3. **Boot Loader Tasks:** The boot loader (e.g., GRUB for Linux) loads the kernel of the operating system into memory.
4. **Kernel Initialization:** The kernel initializes the system's hardware components and starts system processes.
5. **System Initialization:** The init system (e.g., systemd for Linux) executes system initialization scripts to set up the environment and start essential services.
6. **Login Prompt:** The system presents the user with a login prompt (CLI or GUI), allowing the user to log in and begin using the system.

Each step in the booting process is crucial for ensuring the operating system loads correctly and the system is ready for user interaction.