Unix commands are a set of commands that are used to interact with the Unix operating system. Unix is a powerful, multi-user, multi-tasking operating system that was developed in the 1960s by Bell Labs. Unix commands are entered at the command prompt in a terminal window, and they allow users to perform a wide variety of tasks, such as managing files and directories, running processes, managing user accounts, and configuring network settings. Unix is now one of the most commonly used Operating systems used for various purposes such as Personal use, Servers, Smartphones, and many more. It was developed in the 1970’s at AT& T Labs by two famous personalities Dennis M. Ritchie and Ken Thompson.

You’ll be surprised to know that the most popular programming language C came into existence to write the Unix Operating System.

Linux is Unix-Like operating system.

The most important part of the Linux is Linux Kernel which was first released in the early 90s by Linus Torvalds. There are several Linux distros available (most are open-source and free to download and use) such as Ubuntu, Debian, Fedora, Kali, Mint, Gentoo, Arch and much more.

Unix and the C were found by AT&T and distributed to government and academic institutions, which led to both being ported to a wider variety of machine families than any other operating system. The main focus that was brought by the developers in this operating system was the Kernel . Unix was considered to be the heart of the operating System. The system Structure of Unix OS are as follows:

UNIX is a family of multitasking, multiuser computer operating systems developed in the mid 1960s at Bell Labs. It was originally developed for mini computers and has since been ported to various hardware platforms. UNIX has a reputation for stability, security, and scalability, making it a popular choice for enterprise-level computing.

For those preparing for exams like GATE , a thorough understanding of operating systems, including Unix, is essential. Our GATE course provides an in-depth exploration of Unix, covering its history, structure, and key concepts that are crucial for the exam

The basic design philosophy of UNIX is to provide simple, powerful tools that can be combined to perform complex tasks. It features a command-line interface that allows users to interact with the system through a series of commands, rather than through a graphical user interface (GUI).

Some of the key features of UNIX include:

Multiuser support: UNIX allows multiple users to simultaneously access the same system and share resources.

Multitasking: UNIX is capable of running multiple processes at the same time.

Shell scripting: UNIX provides a powerful scripting language that allows users to automate tasks.

Security: UNIX has a robust security model that includes file permissions, user accounts, and network security features.

Portability: UNIX can run on a wide variety of hardware platforms, from small embedded systems to large mainframe computers.

Communication: UNIX supports communication methods using the write command, mail command, etc.

Process Tracking: UNIX maintains a record of the jobs that the user creates. This function improves system performance by monitoring CPU usage. It also allows you to keep track of how much disk space each user uses, and the use that information to regulate disk space.

Today, UNIX is widely used in enterprise-level computing, scientific research, and web servers. Many modern operating systems, including Linux and macOS, are based on UNIX or its variants.

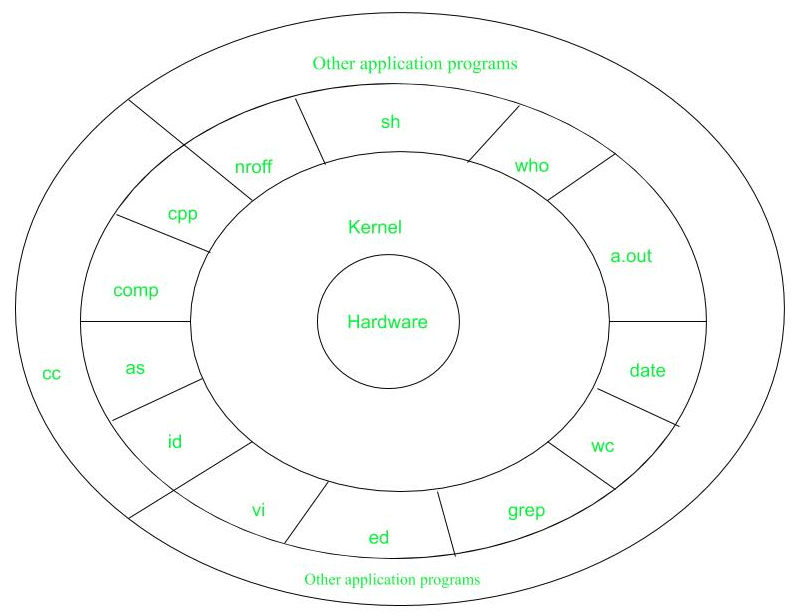


Figure – system structure

Layer-1: Hardware: It consists of all hardware related information.

Layer-2: Kernel: This is the core of the Operating System. It is a software that acts as the interface between the hardware and the software. Most of the tasks like memory management, file management, network management, process management, etc., are done by the kernel.

Layer-3: Shell commands: This is the interface between the user and the kernel. Shell is the utility that processes your requests. When you type in a command at the terminal, the shell interprets the command and calls the program that you want. There are various commands like cp, mv, cat, grep, id, wc, nroff, a.out and more.

Layer-4: Application Layer: It is the outermost layer that executes the given external applications.

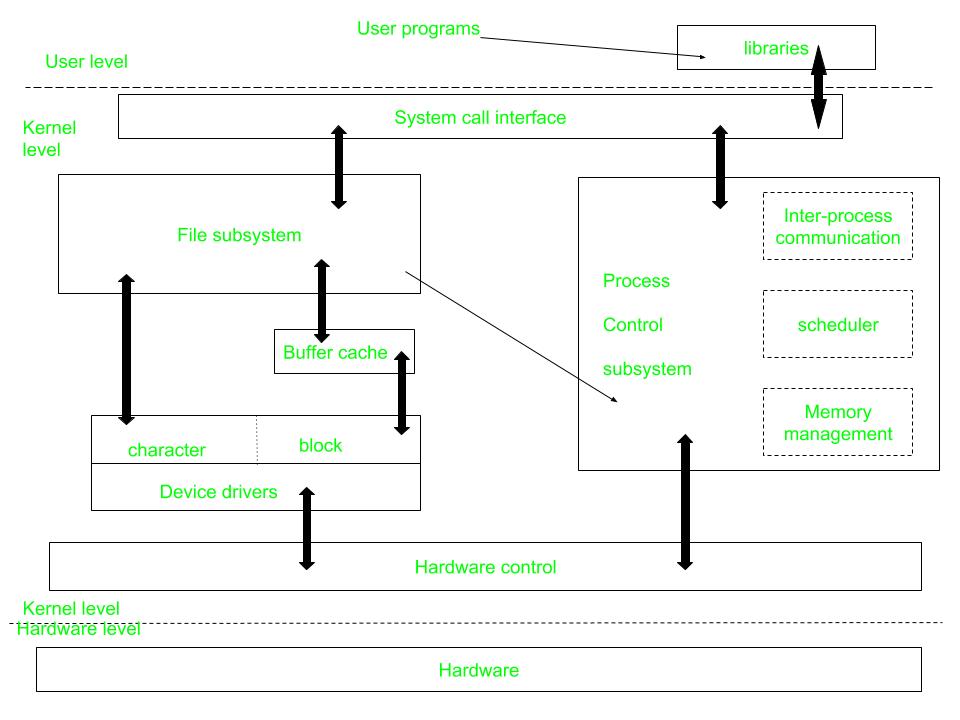


Figure – kernel and its block diagram

This diagram shows three levels: user, kernel, and hardware.

The system call and library interface represent the border between user programs and the kernel. System calls look like ordinary function calls in C programs. Assembly language programs may invoke system calls directly without a system call library. The libraries are linked with the programs at compile time.

The set of system calls into those that interact with the file subsystem and some system calls interact with the process control subsystem. The file subsystem manages files, allocating file space, administering free space, controlling access to files, and retrieving data for users.

Processes interact with the file subsystem via a specific set of system calls, such as open (to open a file for reading or writing), close, read, write, stat (query the attributes of a file), chown (change the record of who owns the file), and chmod (change the access permissions of a file).

The file subsystem accesses file data using a buffering mechanism that regulates data flow between the kernel and secondary storage devices. The buffering mechanism interacts with block I/O device drivers to initiate data transfer to and from the kernel.

Device drivers are the kernel modules that control the operator of peripheral devices. The file subsystem also interacts directly with “raw” I/O device drivers without the intervention of the buffering mechanism. Finally, the hardware control is responsible for handling interrupts and for communicating with the machine. Devices such as disks or terminals may interrupt the CPU while a process is executing. If so, the kernel may resume execution of the interrupted process after servicing the interrupt.

Interrupts are not serviced by special processes but by special functions in the kernel, called in the context of the currently running process.

**Difference between Unix and Linux**

Linux is essentially a clone of Unix. But, basic differences are shown below:

| **Linux** | **Unix** |
| --- | --- |
| The source code of Linux is freely available to its users | The source code of Unix is not freely available general public |
| It has graphical user interface along with command line interface | It only has command line interface |
| Linux OS is portable, flexible, and can be executed in different hard drives | Unix OS is not portable |
| Different versions of Linux OS are Ubuntu, Linux Mint, RedHat Enterprise Linux, Solaris, etc. | Different version of Unix are AIS, HP-UX, BSD, Iris, etc. |
| The file systems supported by Linux are as follows: xfs, ramfs, vfat, cramfsm, ext3, ext4, ext2, ext1, ufs, autofs, devpts, ntfs | The file systems supported by Unix are as follows: zfs, js, hfx, gps, xfs, vxfs |
| Linux is an open-source operating system that was first released in 1991 by Linus Torvalds. | Unix is a proprietary operating system that was originally developed by AT&T Bell Labs in the mid 1960s. |
| The Linux kernel is monolithic, meaning that all of its services are provided by a single kernel. | The Unix kernel is modular, meaning that it is made up of a collection of independent modules that can be loaded and unloaded dynamically. |
| Linux has much broader hardware support than Unix. | Unix was originally designed to run on large, expensive mainframe computers, while Linux was designed to run on commodity hardware like PCs and servers. |
| User Interface of Linux is Graphical or text-based. | User Interface of unix is text-based. |
| Command Line Interface of Linux is Bash, Zsh, Tcsh. | Command Line Interface of unix is Bourne, Korn, C, Zsh. |

Advantages of UNIX:

Stability: UNIX is known for its stability and reliability. It can run for long periods of time without requiring a reboot, which makes it ideal for critical systems that need to run continuously.

Security: UNIX has a robust security model that includes file permissions, user accounts, and network security features. This makes it a popular choice for systems that require high levels of security.

Scalability: UNIX can be scaled up to handle large workloads and can be used on a variety of hardware platforms.

Flexibility: UNIX is highly customizable and can be configured to suit a wide range of needs. It can be used for everything from simple desktop systems to complex server environments.

Command-line interface: UNIX’s command-line interface allows for powerful and efficient interaction with the system.

Disadvantages of UNIX:

Complexity: UNIX can be complex and difficult to learn for users who are used to graphical user interfaces (GUIs).

Cost: Some UNIX systems can be expensive, especially when compared to open-source alternatives like Linux.

Lack of standardization: There are many different versions of UNIX, which can make it difficult to ensure compatibility between different systems.

Limited software availability: Some specialized software may not be available for UNIX systems.

Steep learning curve: UNIX requires a certain level of technical knowledge and expertise, which can make it challenging for novice users.

--------------------------------------------------------------------------------

Now coming to the Basic and most usable commands of Linux/Unix part. (Please note that all the linux/unix commands are run in the terminal of a linux system.Terminal is like command prompt as that of in Windows OS)

Linux/Unix commands are case-sensitive i.e Hello is different from hello.

Basic Unix commands:

Table of Content:-

File System Navigation Unix Command

File Manipulation Unix Command

Process Management Unix Command

Text Processing Unix Command

Network Communication Unix Command

Text Editors in Unix

**File System Navigation Unix Command**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| **cd** | Changes the current working directory. | cd Documents |
| **ls** | Lists files and directories in the current directory. | ls |
| **pwd** | Prints the current working directory. | pwd |
| **mkdir** | Creates a new directory. | mkdir new\_folder |
| **rmdir** | Removes an empty directory. | rmdir empty\_folder |
| **mv** | Moves files or directories. | mv file1.txt Documents/ |

**File Manipulation Unix Command**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| **touch** | Creates an empty file or updates the access and modification times. | touch new\_file.txt |
| **cp** | Copies files or directories. | cp file1.txt /opt/file2.txt |
| **mv** | Moves files or directories. | mv file1.txt Documents |
| **rm** | Remove files or directories. | rm old\_file.txt |
| **chmod** | Changes the permissions of a file or directory. | chmod 644 file.txt |
| **chown** | Changes the owner and group of a file or directory. | chown user:group file.txt |
| **ln** | Creates links between files. | ln -s target\_file symlink |
| **cat** | Concatenates files and displays their contents. | cat file1.txt file2.txt |
| **head** | Displays the first few lines of a file. | head file.txt |
| **tail** | Displays the last few lines of a file. | tail file.txt |
| **more** | Displays the contents of a file page by page. | more file.txt |
| **less** | Displays the contents of a file with advanced navigation features. | less file.txt |
| **diff** | Compares files line by line. | diff file1.txt file2.txt |
| **patch** | Applies a diff file to update a target file. | patch file.txt < changes.diff |

**Process Management Unix Command**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| **ps** | Displays information about active processes, including their status and IDs. | ps aux |
| **top** | Displays a dynamic real-time view of system processes and their resource usage. | top |
| **kill** | Terminates processes using their process IDs (PIDs). | kill <pid> |
| **pkill** | Sends signals to processes based on name or other attributes. | pkill -9 firefox |
| **killall** | Terminates processes by name. | killall -9 firefox |
| **renice** | Changes the priority of running processes. | renice -n 10 <pid> |
| **nice** | Runs a command with modified scheduling priority. | nice -n 10 command |
| **pstree** | Displays running processes as a tree. | pstree |
| **pgrep** | Searches for processes by name or other attributes. | pgrep firefox |
| **jobs** | Lists active jobs and their status in the current shell session. | jobs |
| **bg** | Puts a job in the background. | bg <job\_id> |
| **fg** | Brings a background job to the foreground. | fg <job\_id> |
| **nohup** | Runs a command immune to hangups, with output to a specified file. | nohup command & |
| **disown** | Removes jobs from the shell’s job table, allowing them to run independently. | disown <job\_id> |

**Text Processing Unix Command**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| **grep** | Searches for patterns in text files. | grep "error" logfile.txt |
| **sed** | Processes and transforms text streams. | sed 's/old\_string/new\_string/g' file.txt |
| **awk** | Processes and analyzes text files using a pattern scanning and processing language. | awk '{print $1, $3}' data.csv |

**Network Communication Unix Command**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| **ping** | Tests connectivity with another host using ICMP echo requests. | ping google.com |
| **traceroute** | Traces the route that packets take to reach a destination. | traceroute google.com |
| **nslookup** | Queries DNS servers for domain name resolution and IP address information. | nslookup google.com |
| **dig** | Performs DNS queries, providing detailed information about DNS records. | dig google.com |
| **host** | Performs DNS lookups, displaying domain name to IP address resolution. | host google.com |
| **whois** | Retrieves information about domain registration and ownership. | whois google.com |
| **ssh** | Provides secure remote access to a system. | ssh username@hostname |
| **scp** | Securely copies files between hosts over a network. | scp file.txt username@hostname:/path/ |
| **ftp** | Transfers files between hosts using the File Transfer Protocol (FTP). | ftp hostname |
| **telnet** | Establishes interactive text-based communication with a remote host. | telnet hostname |
| **netstat** | Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships. | netstat -tuln |
| **ifconfig** | Displays or configures network interfaces and their settings. | ifconfig |
| **iwconfig** | Configures wireless network interfaces. | iwconfig wlan0 |
| **route** | Displays or modifies the IP routing table. | route -n |
| **arp** | Displays or modifies the Address Resolution Protocol (ARP) cache. | arp -a |
| **ss** | Displays socket statistics. | ss -tuln |
| **hostname** | Displays or sets the system’s hostname. | hostname |
| **mtr** | Combines the functionality of ping and traceroute, providing detailed network diagnostic information. | mtr google.com |

**System Administration Unix Command**

| **Command** | **Description** | **Example** |
| --- | --- | --- |
| **df** | Displays disk space usage. | df -h |
| **du** | Displays disk usage of files and directories. | du -sh /path/to/directory |
| **crontab -e** | Manages cron jobs, which are scheduled tasks that run at predefined times or intervals. | crontab -e |

**Text Editors in Unix**

| **Text Editor** | **Description** | **Example** |
| --- | --- | --- |
| **Vi / Vim** | Vi (Vim) is a highly configurable, powerful, and feature-rich text editor based on the original Vi editor. Vim offers modes for both command-line operations and text editing. | Open a file with Vim: vim filename Exit Vim editor: Press Esc, then type :wq and press Enter |
| **Emacs** | Emacs is a versatile text editor with extensive customization capabilities and support for various programming languages. | Open a file with Emacs: emacs filename Save and exit Emacs: Press Ctrl + X, then Ctrl + S and Ctrl + X, then Ctrl + C to exit |
| **Nano** | Nano is a simple and user-friendly text editor designed for ease of use and accessibility. | Open a file with Nano: nano filename Save and exit Nano: Press Ctrl + O, then Ctrl + X |
| **Ed** | Ed is a standard Unix text editor that operates in line-oriented mode, making it suitable for batch processing and automation tasks. | Open a file with Ed: ed filename Exit Ed editor: Type q and press Enter |
| **Jed** | Jed is a lightweight yet powerful text editor that provides an intuitive interface and support for various programming languages. | Open a file with Jed: jed filename Save and exit Jed: Press Alt + X, then type exit and press Enter |

Unix Commands – FAQs

**What is Unix and how does it differ from other operating systems?**

This question aims to clarify the unique features and characteristics of Unix compared to other operating systems like Windows or macOS.

**Who developed Unix and what is its significance in the history of computing?**

Users might want to know about the origins of Unix, its developers, and its role in shaping the modern computing landscape.

**What are some popular Unix-like operating systems and how do they relate to Unix?**

This question seeks to understand the relationship between Unix and Unix-like systems such as Linux, and the various distributions available for different purposes.

**What are the essential Unix commands and how are they used?**

Users may seek clarification on the basic Unix commands listed in the article and how they can be applied in practical scenarios.

**Conclusion**

In conclusion, Unix commands serve as a fundamental toolkit for navigating and managing the Unix operating system, which has evolved from its inception in the 1960s to become one of the most widely used OS platforms across various domains including personal computing, servers, and mobile devices. From its origins at Bell Labs with developers Dennis M. Ritchie and Ken Thompson to the birth of the C programming language and the subsequent emergence of Unix-like systems such as Linux, the Unix ecosystem has significantly shaped the computing landscape. Understanding basic Unix commands is essential for users to efficiently manipulate files, manage processes, configure networks, and perform system administration tasks, thereby empowering them to leverage the full potential of Unix-based systems for diverse computing needs.