Department of Computer Science & Information Technology

III Year, V Semester (Batch 2022-2026)

Lab Record Submission of

Linux (Lab)

Subject Code – CSIT-505

Submitted to: Submitted by:

Prof. Nidhi Nigam Tilak Neema

(0827CI221138)

**Name: Tilak Neema Enrollment No: 0827CI221138**

**Course Name: LINUX Subject Code: CSIT-505**

**Subject Name: LINUX LAB**

**INDEX PAGE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Name** | **Experiment Date** | **Submission Date** | **Page No.** | **Sign** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Introduction to Operating Systems**

**An Operating System (OS) acts as an interface between the user and the computer hardware, managing the system's resources and offering essential services for applications. Its core responsibilities include:**

* **Process Management: Handles the creation, scheduling, and termination of processes, ensuring smooth execution.**
* **Memory Management: Oversees the allocation and deallocation of memory for various programs, preventing memory conflicts.**
* **File Management: Controls the organization, storage, and retrieval of data, ensuring efficient file handling.**
* **Device Management: Manages input/output devices, facilitating communication between hardware components.**
* **Security and Access Control: Safeguards system resources and data from unauthorized access, enforcing security policies.**

**Operating systems are crucial for maintaining the smooth operation of a computer, creating a stable environment for applications to function effectively.**

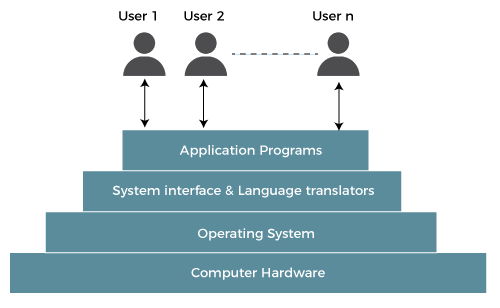


Figure 1Operating system flow

**Services Provided by Operating Systems**

**Operating systems offer a range of services to both users and applications, enabling efficient operation and interaction:**

* **User Interface: Provides both command-line and graphical interfaces for user interaction.**
* **Program Execution: Loads and runs applications, ensuring they have access to necessary resources.**
* **I/O Operations: Manages input/output activities like reading/writing to disks or interacting with peripherals.**
* **Error Detection and Handling: Monitors for potential errors and implements mechanisms to manage and recover from them.**
* **Resource Allocation: Distributes essential resources such as CPU cycles, memory, and storage to different processes as needed.**

**Why Operating Systems Are Essential**

**Operating systems are a vital component in computer systems for several reasons:**

* **Resource Management: They effectively manage and allocate system resources, ensuring optimal performance and utilization.**
* **User Convenience: They simplify complex hardware operations, making it easier for users to interact with the system.**
* **System Security: By enforcing security protocols, they protect data and resources from unauthorized access and potential threats.**

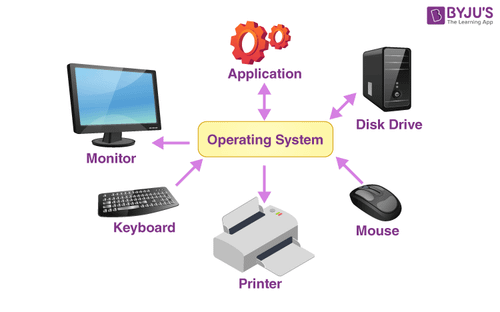


Figure 2 Operating system devices

**Linux: Introduction**

**Linux is a widely-used, free, and open-source operating system built on the principles of Unix. Known for its stability and robust security features, Linux is commonly deployed in servers, desktops, and embedded systems. Linux shares many functionalities with Unix, ensuring compatibility while offering a modern, customizable platform.**

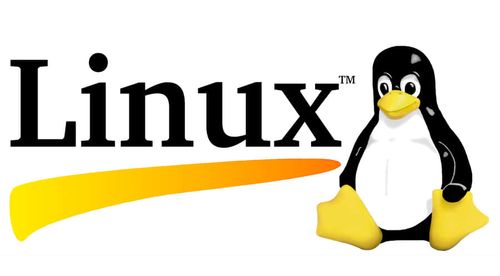


Figure 3 Linux Logo

**Uses:**

**Linux offers several advantages over other operating systems:**

* **Open Source: As an open-source platform, Linux allows users to view, modify, and distribute its source code, fostering collaboration and innovation in the software community.**
* **Cost-Effective: Linux is free to use, eliminating the need for costly software licenses and reducing expenses for both individuals and businesses.**
* **Stability and Security: It’s highly secure and stable, making it suitable for servers and critical systems where uptime and data protection are paramount.**

**Key Services Provided by Linux**

**Like other operating systems, Linux provides a variety of services that enhance its usability and versatility:**

* **Multi-user Capabilities: Multiple users can access the system simultaneously without interfering with each other’s tasks.**
* **Multitasking: Linux allows several processes to run concurrently, efficiently managing system resources.**
* **Networking Support: Linux is known for its strong networking capabilities, making it an excellent choice for building servers and connecting with other systems.**

**History of Linux**

**Linux has a rich history that traces back to its creation by Linus Torvalds in 1991 as a personal project to develop a free operating system kernel. Since then, Linux has evolved into a global platform used in a variety of fields. Major milestones in Linux history include:**

* **1991: Linus Torvalds released the first version of the Linux kernel.**
* **1992: Linux became a fully operational operating system as critical features were added.**
* **1996: The emergence of the first commercial Linux distributions led to its widespread adoption, particularly in the server and enterprise environments.**

**Popular Linux Distributions**

**Linux is available in numerous distributions (distros), each tailored to different use cases. Some well-known Linux distributions are:**

* **Ubuntu: One of the most popular desktop Linux distributions, known for its user-friendliness and ease of use.**
* **CentOS: A distribution focused on stability and reliability, often used for enterprise servers.**
* **Debian: Famous for its robustness and large software repository, Debian is a popular choice for both desktop and server use.**
* **Fedora: Known for featuring cutting-edge technologies, Fedora is often used by developers and those looking to experiment with the latest software innovations.**

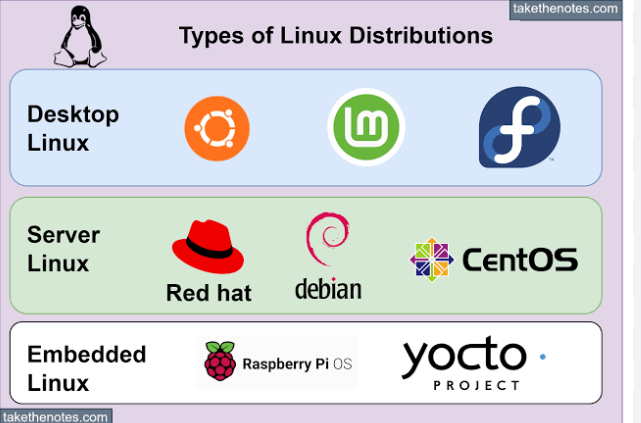


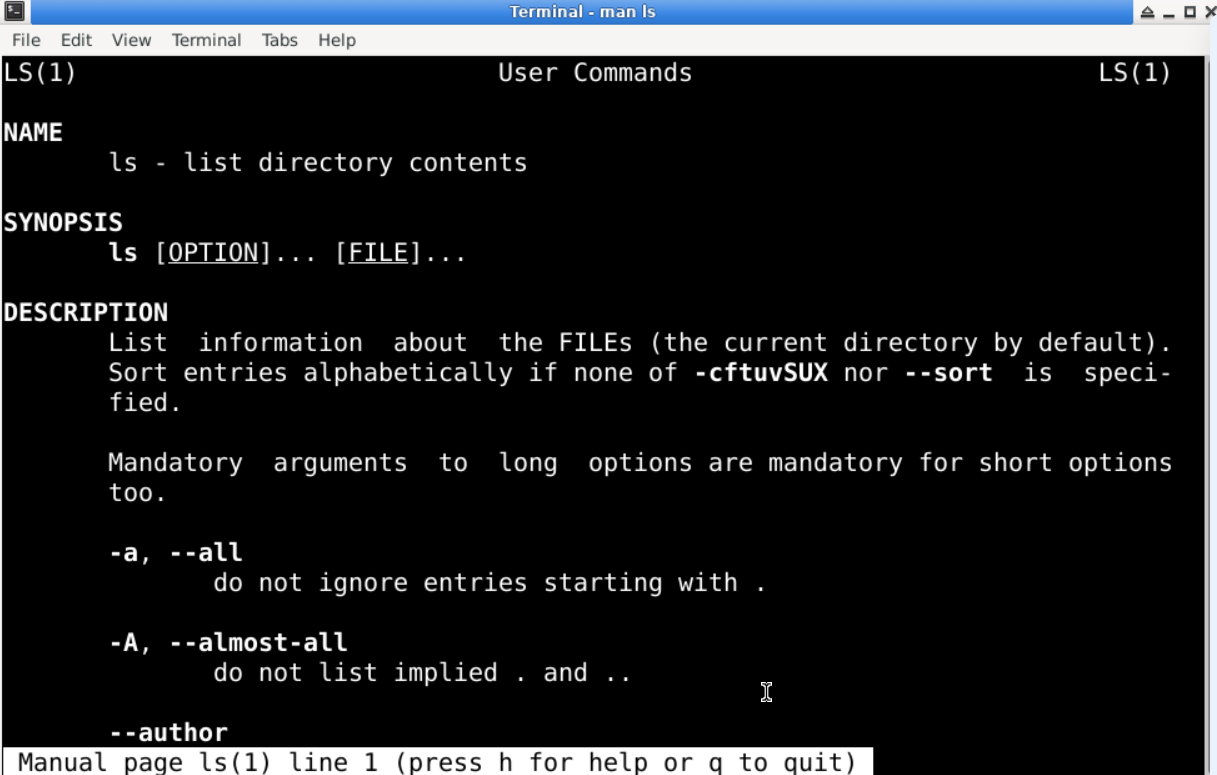
Figure 4 Linux Distribution

**Linux Commands**

* **Man command**

Syntax- man (common name)

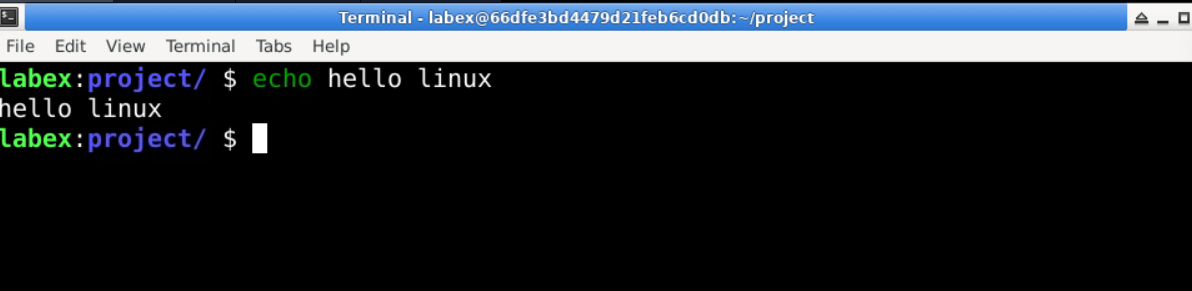
Description- This is the command itself, which invokes the manual page viewer.



* **echo command**

Syntax- echo(statement)

Description: This is a built-in command in many shell environments, including Bash. Its primary function is to display a line of text or a variable value to the standard output (usually the terminal).



* **whatis command**

Syntax- whatis(command name)

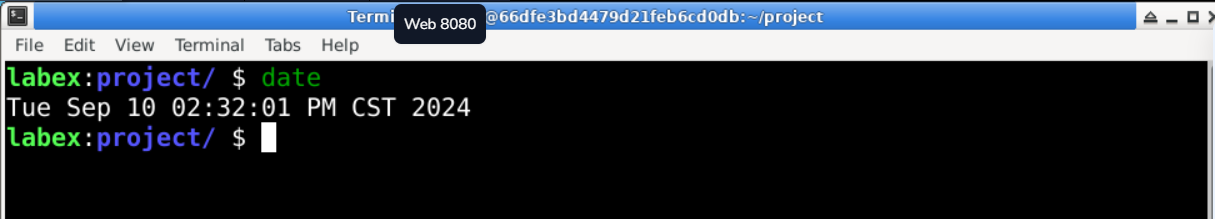
Description : This is the command itself, which queries the manual database for descriptions of commands.



* **date command**

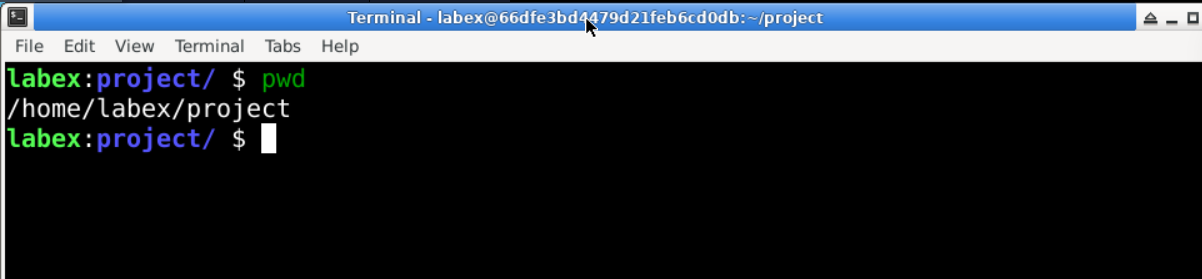
Syntax- date (command name)

Description- : This is the command itself, which retrieves and displays the current date and time.



* **pwd command**

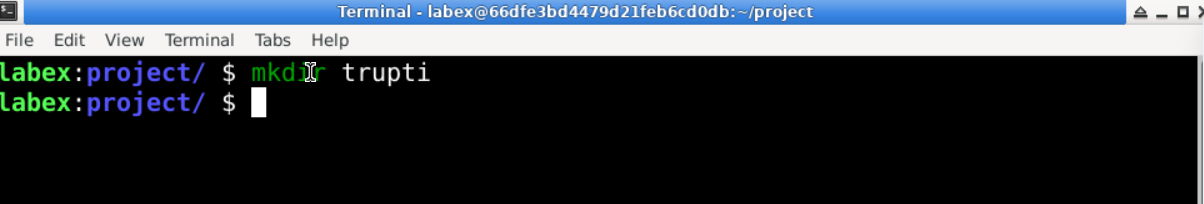
Syntax- pwd

Description- : This is the command itself, which outputs the full path of the current workingdirectory. 

* **mkdir command**

Syntax- mkdir (directory name)

Description- : This is the command itself, which is used to create one or more directories.



* **cd command**

Syntax- cd (directory name)

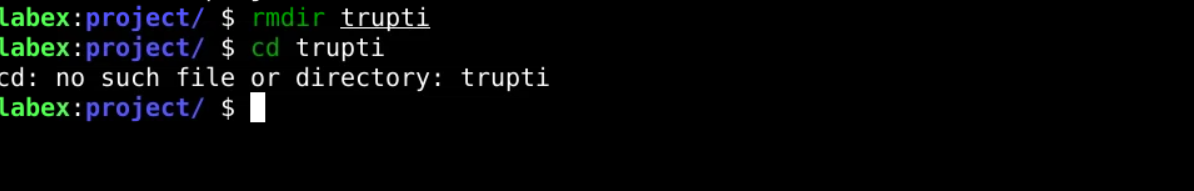
Description- : This is the command itself, which is used to change the current working directory.



* **rmdir command**

Syntax- rmdir (directory name)

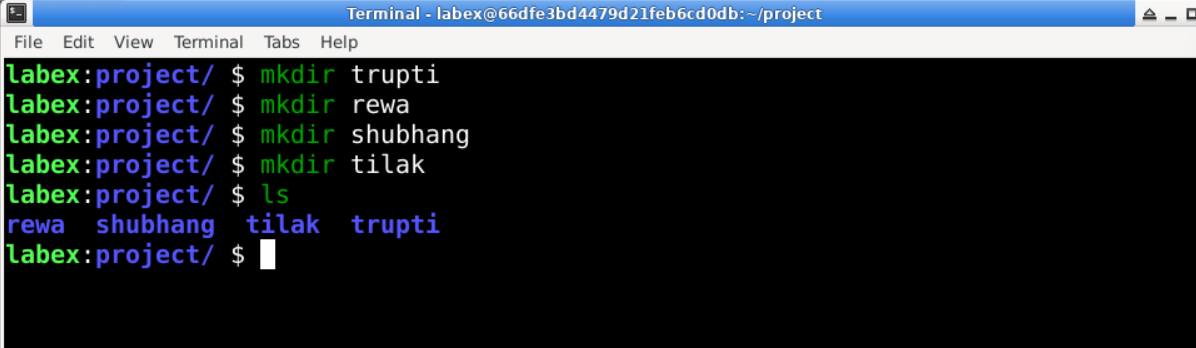
Description- :  This is the command itself, which is used to remove empty directories.



* **ls command**

Syntax- ls

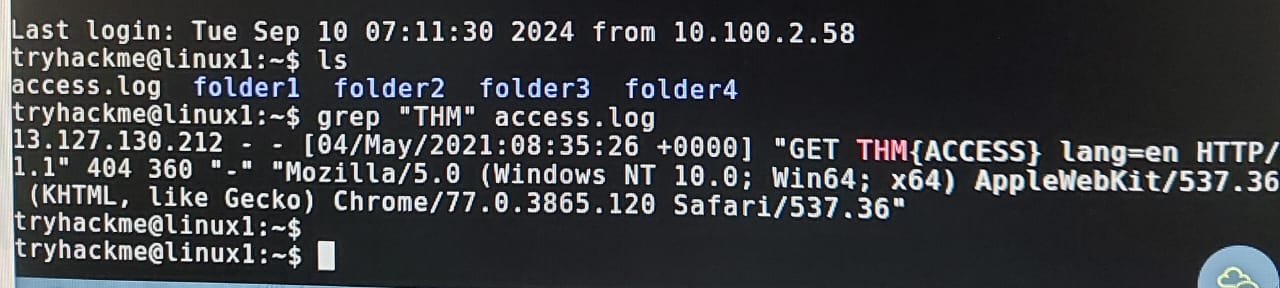
Description- This is the command itself, which stands for "list." By default, it lists the files and directories in the current working directory.



* **grep command**

Syntax- grep(file name)

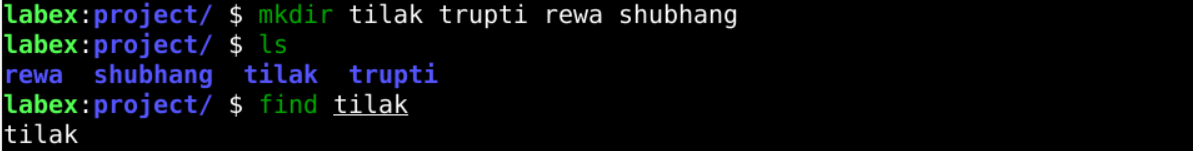
Description- This is the command itself, which searches for patterns in text.



* **find command**

Syntax- find(file name)

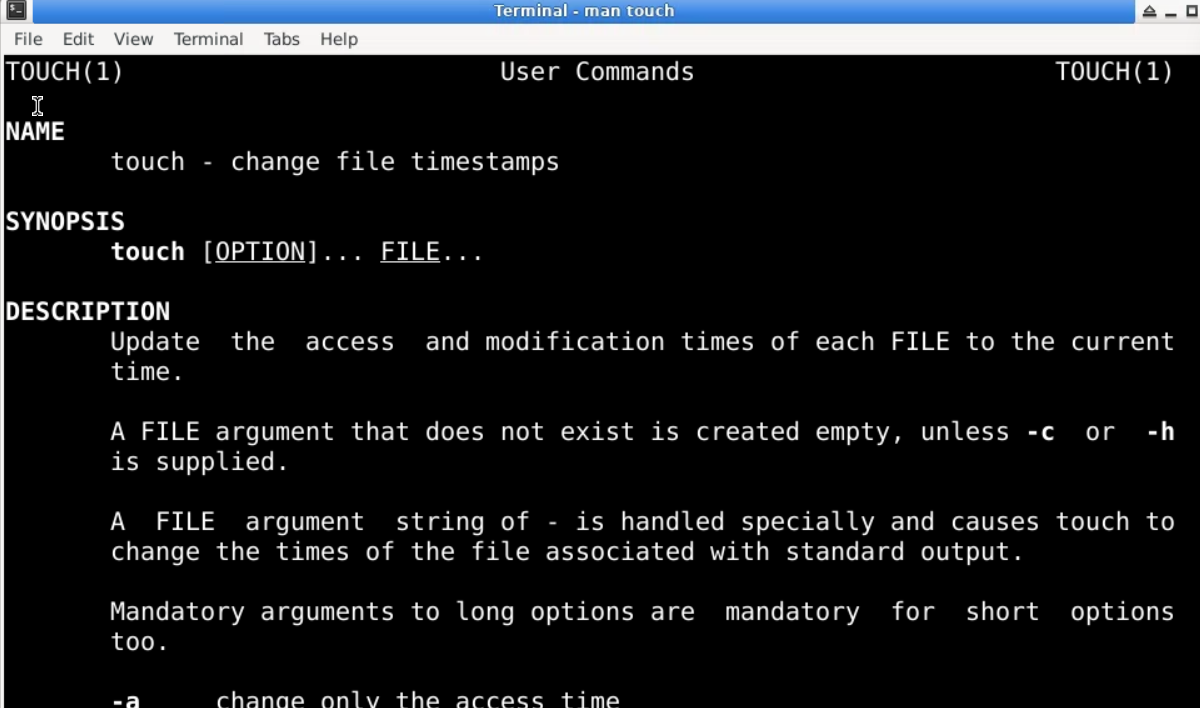
Description- The find command is a powerful utility in Unix/Linux systems used to search for files and directories within a specified directory hierarchy



* **touch command**

Syntax- touch(file name)

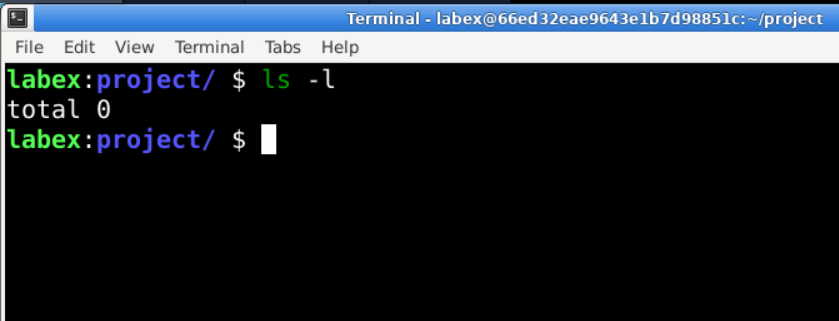
Description- The touch command in Linux is used to create empty files or update the timestamps of existing files



* **ls -l command**

Syntax- ls -l

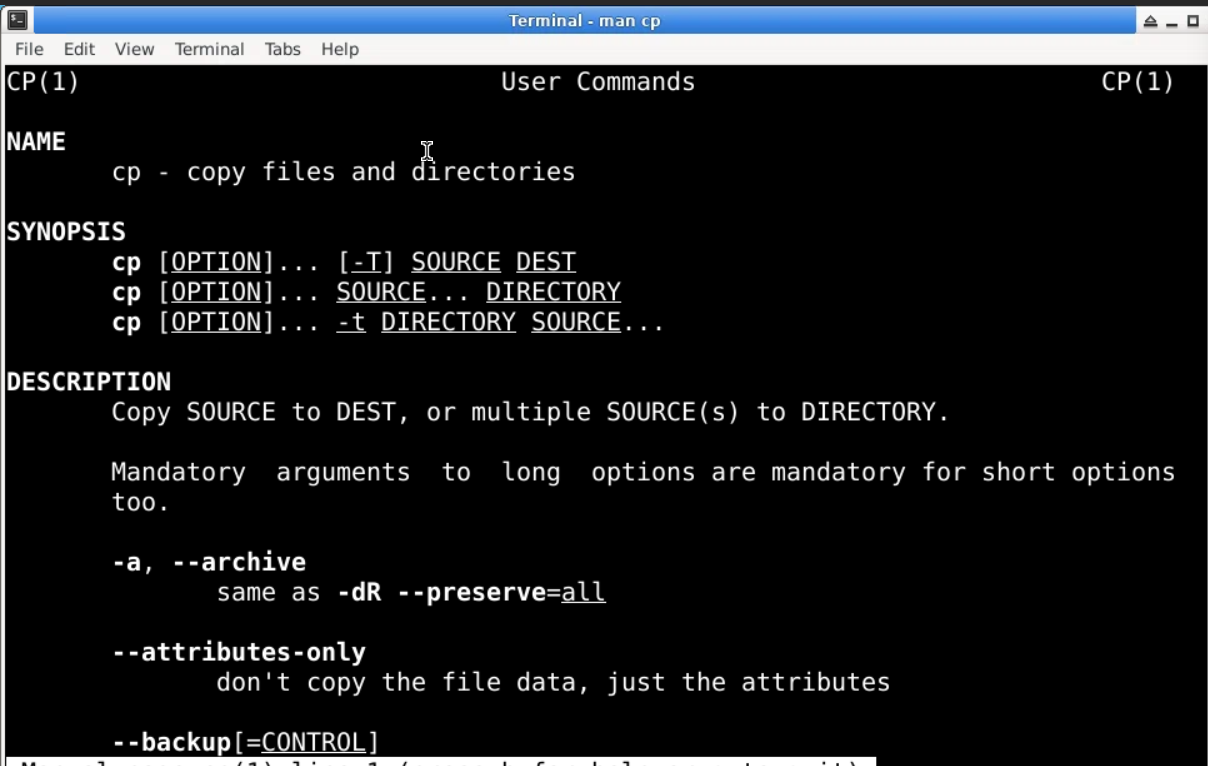
Description- The ls -l command in Linux is used to list files and directories in a detailed (long) format.



* **cp command**

Syntax- cp [options] source destination

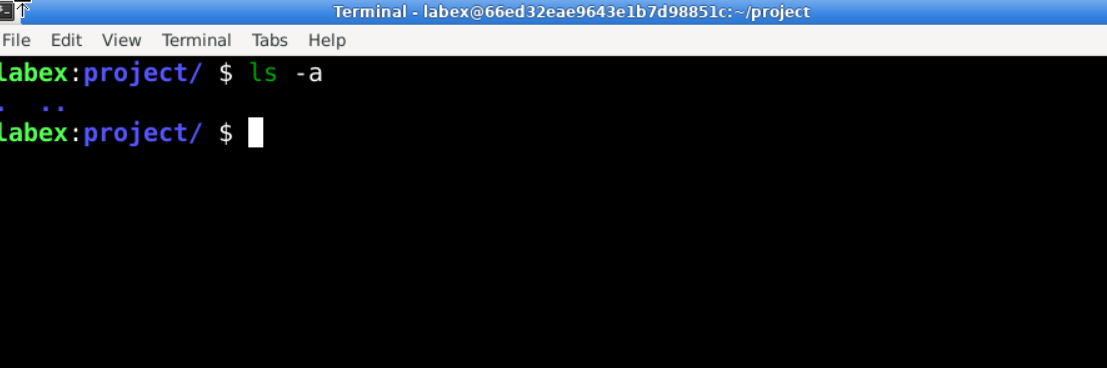
Description- The cp command is a utility in Unix-like operating systems used to copy files and directories.



* **ls -a command**

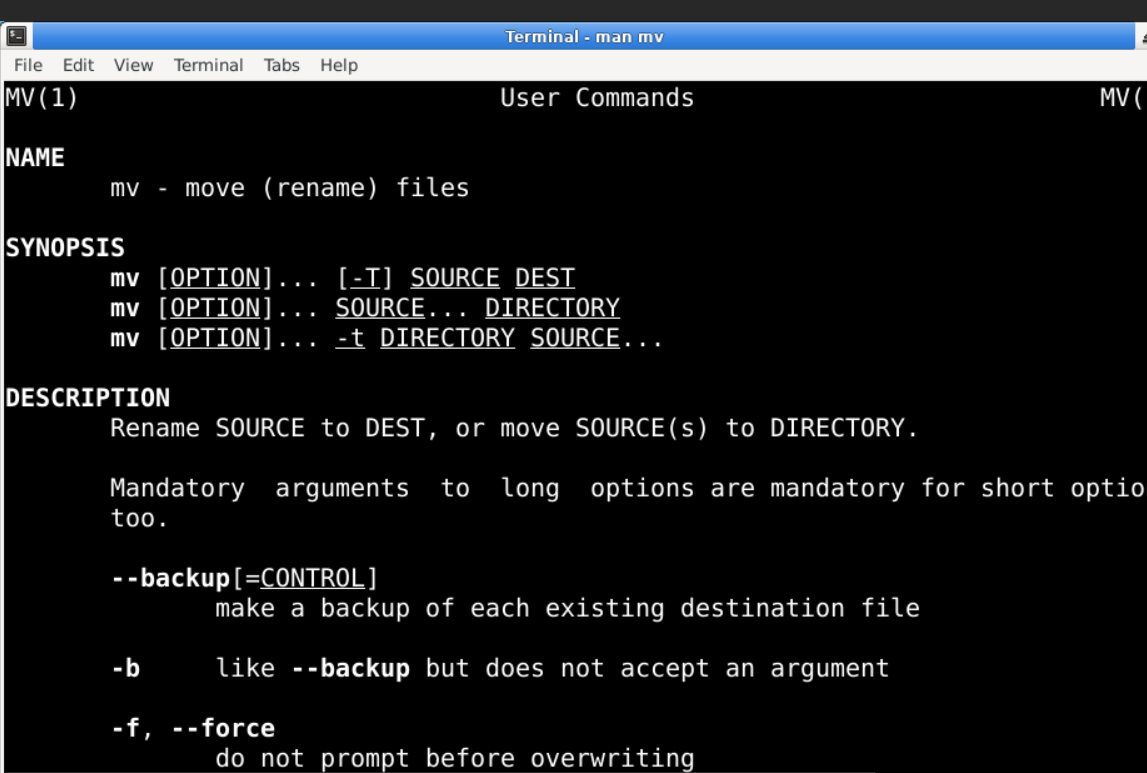
Syntax – ls -a

Description - The ls -a command in Linux is used to list all files and directories in the current directory, including hidden files.



* **mv command**

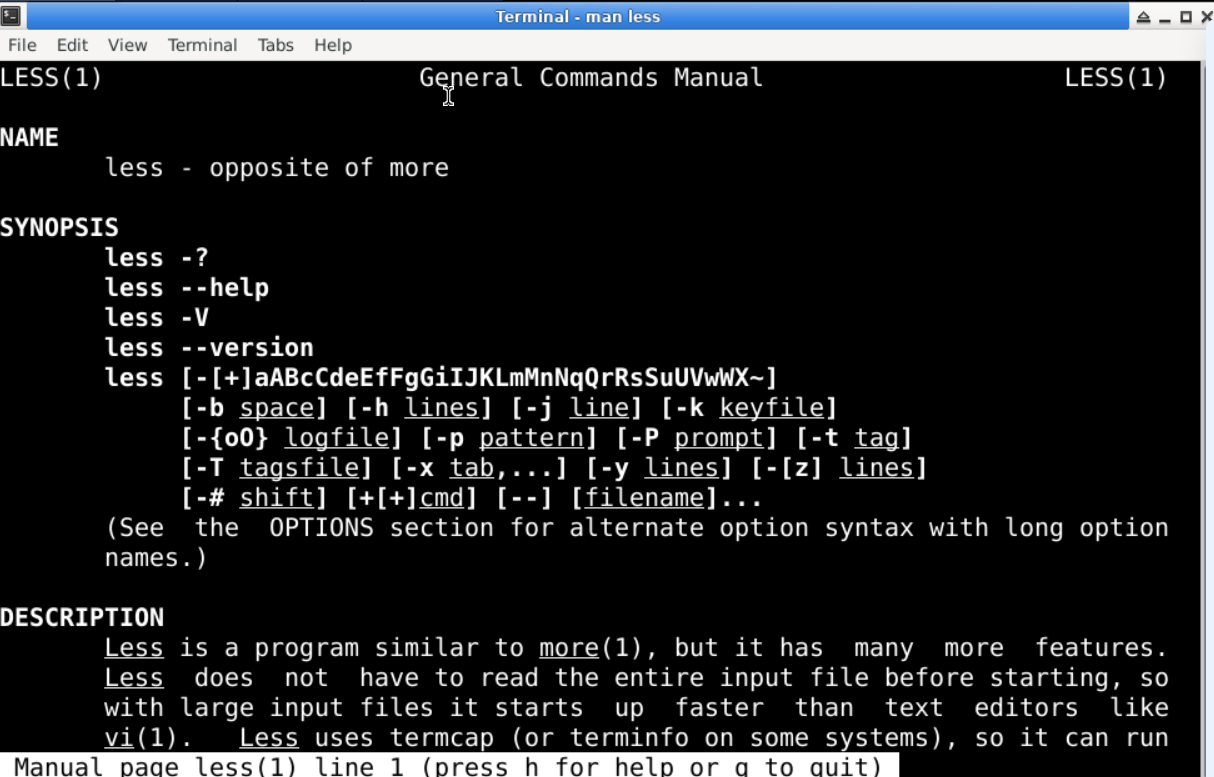
Syntax- mv (file name)

Description- The mv command in Linux is used to move or rename files and direc

* **less command**

Syntax- less (file name)

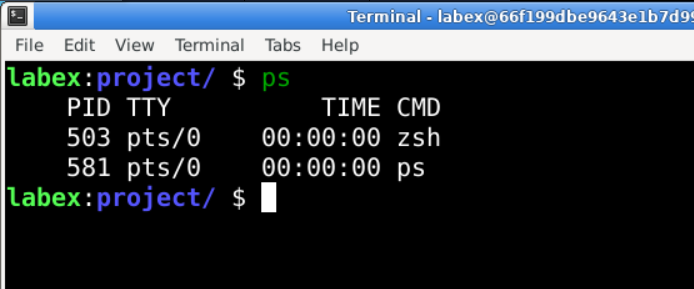
Description- The less command is a commonly used command-line utility in Unix-like operating systems for viewing the contents of files one screen at a time.



* **ps command**

syntax- ps

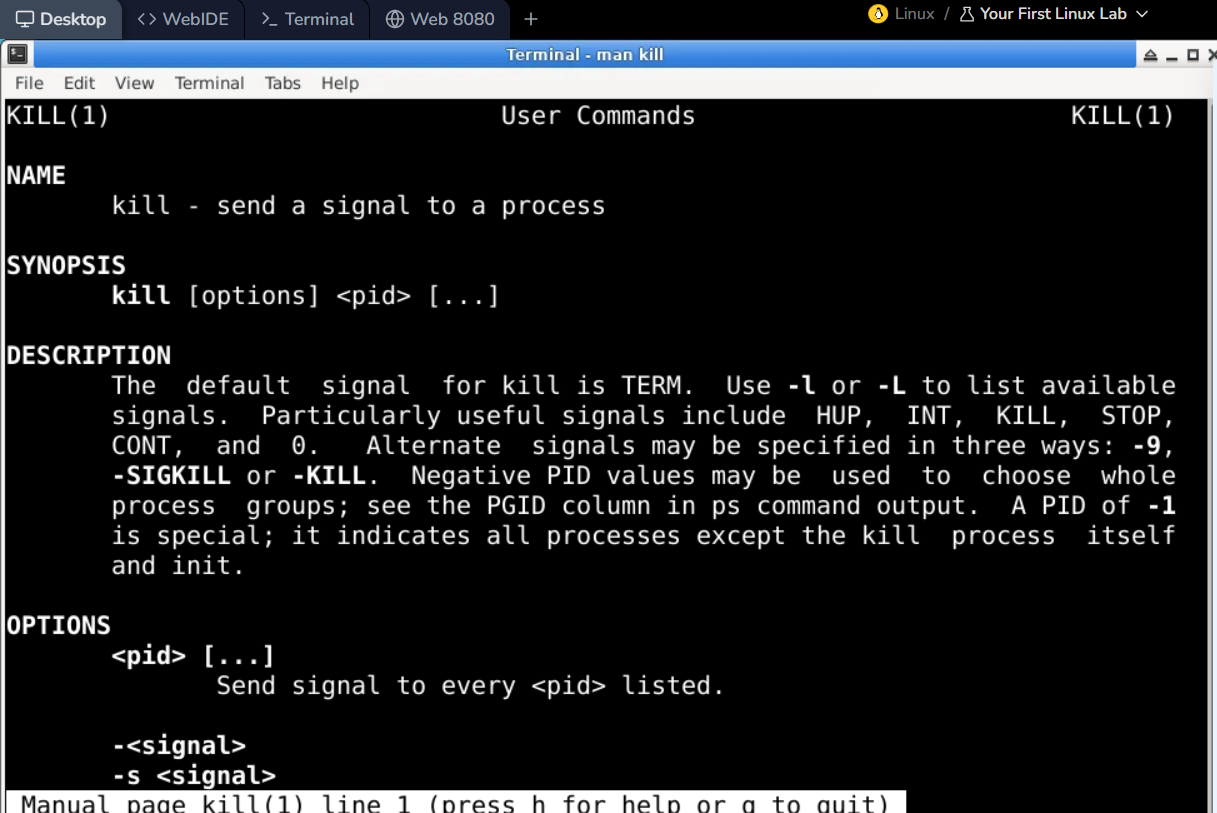
description- The ps command is used in Unix-like operating systems to display information about the currently running processes.



* **kill command**

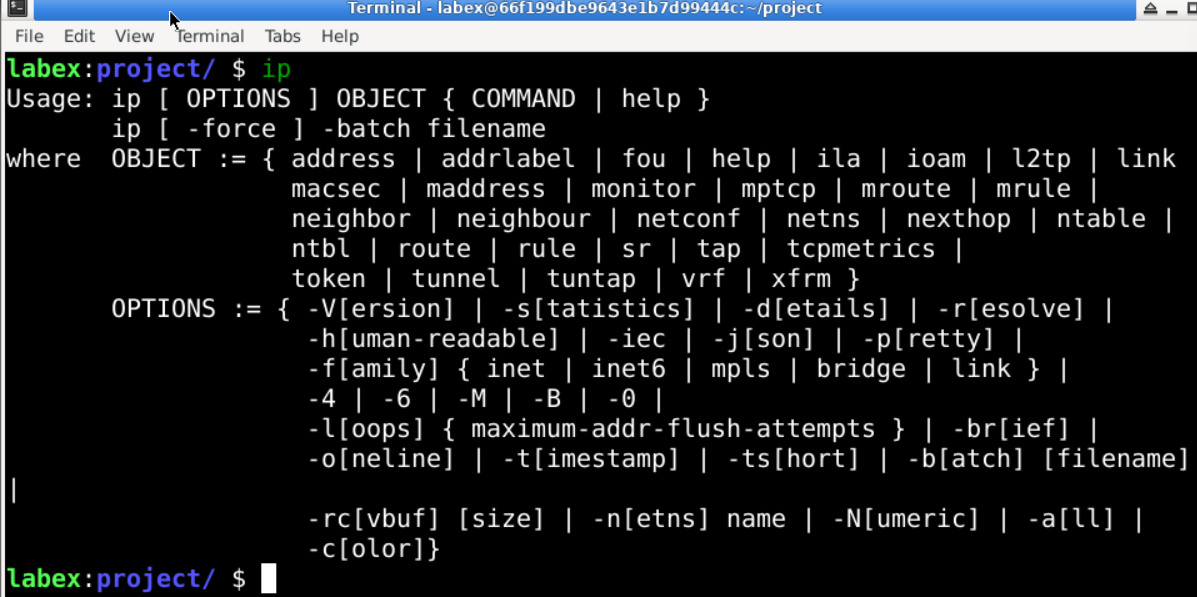
syntax- kill [optin] <pid>

description- The kill command is used in Unix-like operating systems to terminate processes



* **ip**

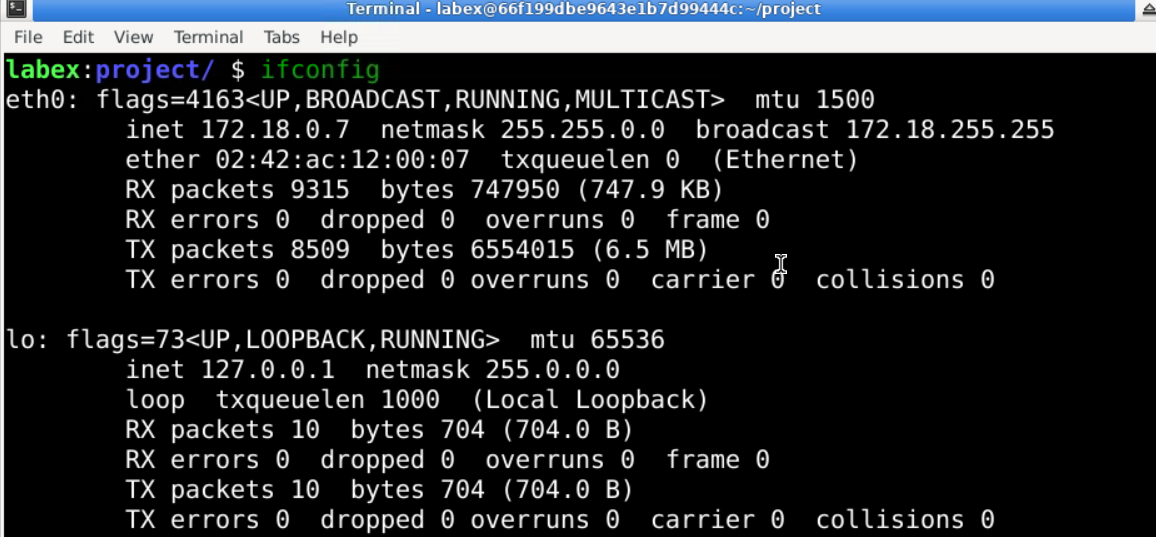
syntax-ip

description- The ip command is a powerful utility in Linux and Unix-like operating systems used for managing network interfaces,

* **ifconfig command**

syntax= ifconfig

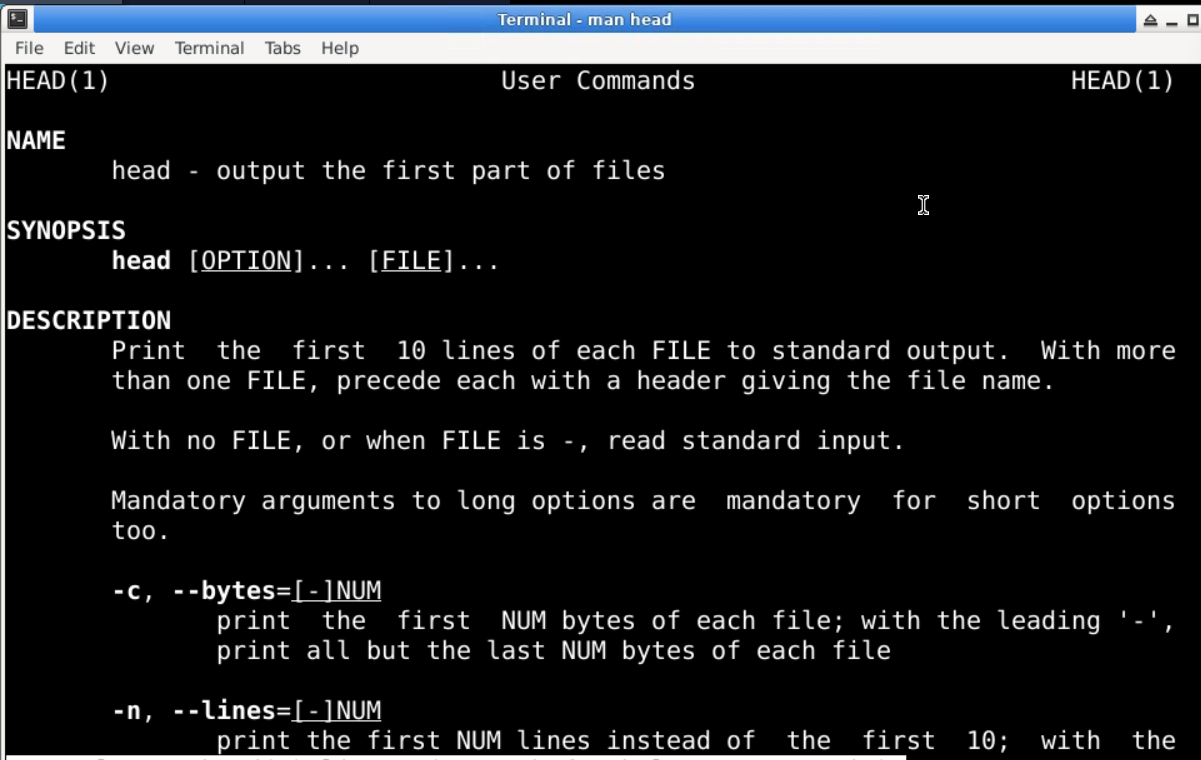
description- The ifconfig command is a network utility in Unix-like operating systems used to configure and display network interface parameters.



* **head command**

syntax- head (file name)

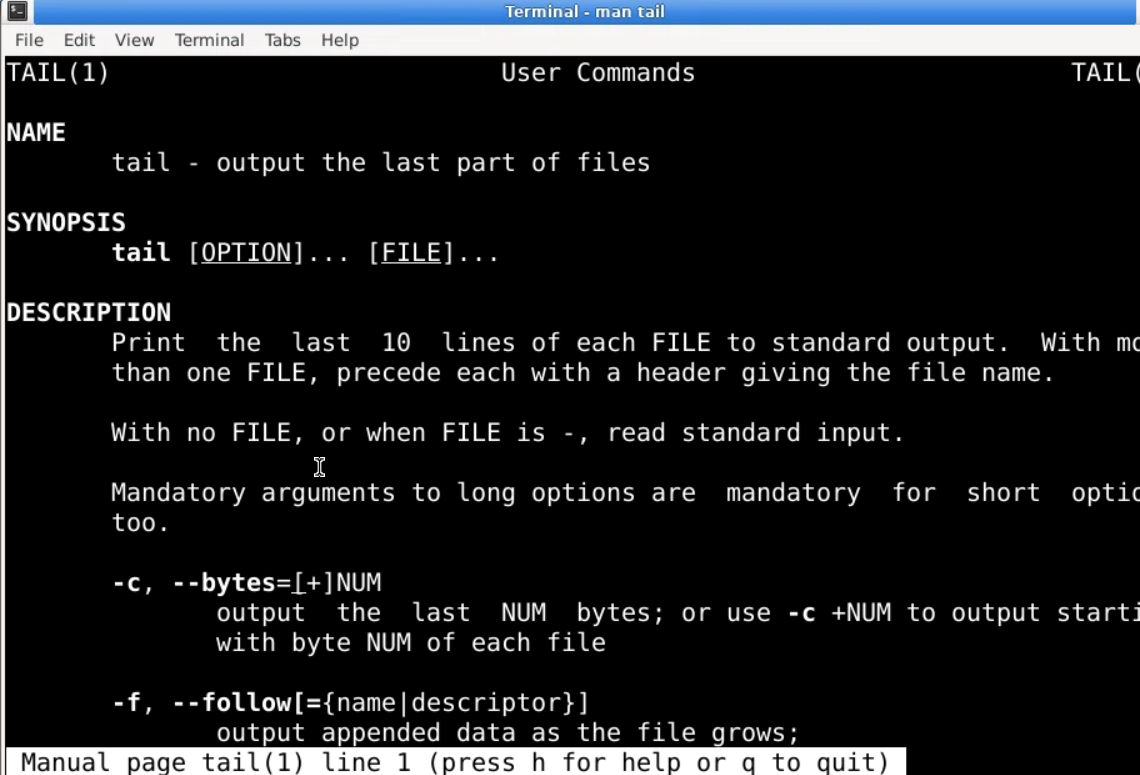
description= The head command is a utility in Unix-like operating systems used to display the beginning of a file or standard input.



* tail command

syntax- tail (filename)

description- The tail command is a utility in Unix-like operating systems used to display the end of a file or standard input



**Linux System Architecture**

The architecture of a Linux system consists of the following layers:

1. **Hardware Layer**  
   This layer consists of all peripheral devices such as RAM, HDD, CPU, and others that provide the foundation for system functionality.
2. **Kernel**  
   The core component of the operating system. It interacts directly with hardware and provides low-level services to the upper layers of the system.
3. **Shell**  
   An interface to the kernel that hides the complexity of kernel functions from the user. It processes user commands and executes kernel functions.
4. **Utilities**  
   Utility programs provide the user with most of the operating system functionalities, ranging from file management to system monitoring.

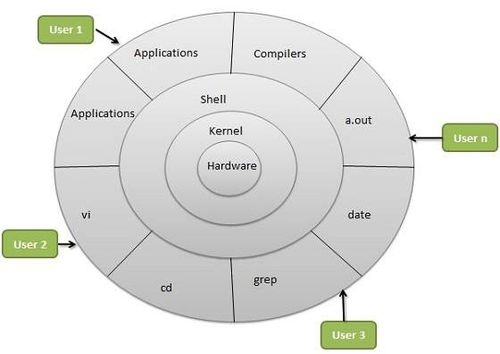
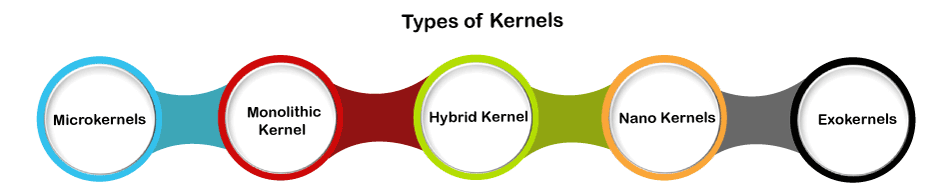


Figure 5Linux System Architecture

**Types of Kernels**  
Kernels are the core of operating systems, and there are various types based on architecture:



1. **Monolithic Kernel**
   * Monolithic kernels include both kernel and user services in the same address space.
   * They use a faster system call communication protocol than microkernels.
   * They are less flexible than microkernels; adding new services requires rebuilding the kernel.
   * Monolithic kernels pose a greater security risk, as failure in one service can bring down the entire system.
   * Example: The Linux kernel uses a monolithic architecture.

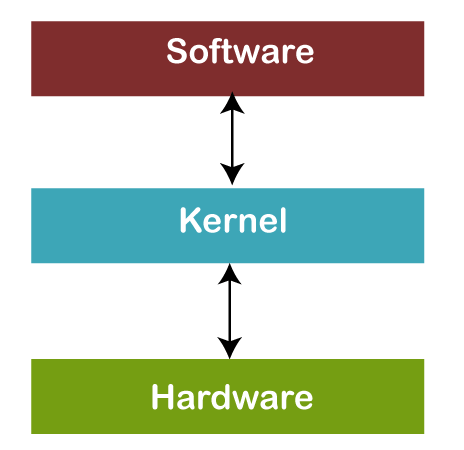


Figure 6 Monolithic Kernel

1. **Microkernels**
   * Microkernels separate kernel services from user processes, delegating them to different address spaces.
   * They use message passing for communication between components.
   * Microkernels are more secure, as failure in one service does not crash the entire system.
   * Example: MINIX 3 is based on a microkernel architecture.

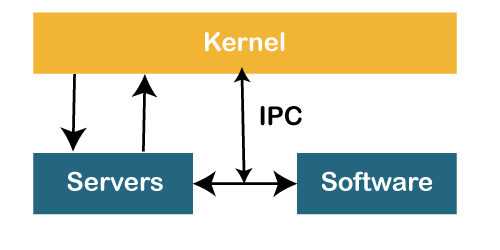


Figure 7 Microkernel

1. **Hybrid Kernel**
   * Combines features of both monolithic and microkernels.
   * Allows modularity with parts of the OS gaining memory protection while maintaining monolithic performance.
   * Example: XNU kernel (used in macOS) is a hybrid of the Mach and BSD kernels.

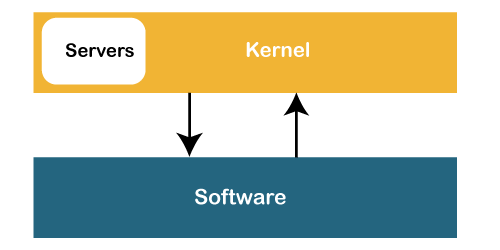


Figure 8 Hybrid Kernel

1. **Exokernel**
   * Exposes hardware resources directly to applications, allowing them more control.
   * Applications can implement their own resource management, leading to performance gains.
   * Example: Exokernels are primarily used in research and experimental operating systems.
2. **Nanokernel**
   * Provides minimal services and leaves most functionality to higher-level components.
   * Focuses on basic hardware management at the lowest level.
3. **Multikernel**
   * Manages hardware resources across multiple kernels.
   * Typically used in distributed systems or environments where different hardware resources need separate management.

**Shell in Linux**  
A shell is a program that allows users to interact with the Linux operating system. The shell interprets commands and interacts with the kernel. Linux provides four commonly used shells:

1. **The C Shell (csh)**
   * Command full-path: /bin/csh
   * Default prompt for a non-root user: hostname %
   * Default prompt for root user: hostname #
2. **The Bourne Shell (sh)**
   * Command full-path: /bin/sh, /sbin/sh
   * Default prompt for a non-root user: $
   * Default prompt for root user: #
3. **The Korn Shell (ksh)**
   * Command full-path: /bin/ksh
   * Default prompt for a non-root user: $
   * Default prompt for root user: #
4. **GNU Bourne-Again Shell (BASH)**
   * Command full-path: /bin/bash
   * Default prompt for a non-root user: bash-g.gg$ (where g.gg indicates the shell version number, e.g., bash-3.50$)
   * Default prompt for root user: bash-g.gg#

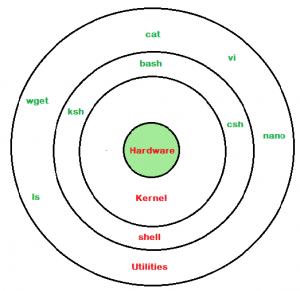


Figure 9 Shell in Linux

* **Hypervisors**  
  Hypervisors are software packages or kernel extensions that allow multiple operating systems to run on a single hardware platform by emulating hardware environments. Examples of hypervisors include:
* VMware Workstation
* VirtualBox.

**Overview of the Linux File System**

**Introduction**

The Linux file system is a crucial component of the Linux operating system, responsible for organizing and managing data on disk drives. Unlike other operating systems, Linux employs a unique hierarchical structure that helps maintain file organization and accessibility.

**File System Structure**

**1. Root Directory (/)**

At the top of the Linux file system hierarchy is the root directory, denoted by /. All other files and directories are organized beneath this root.

**2. Key Directories**

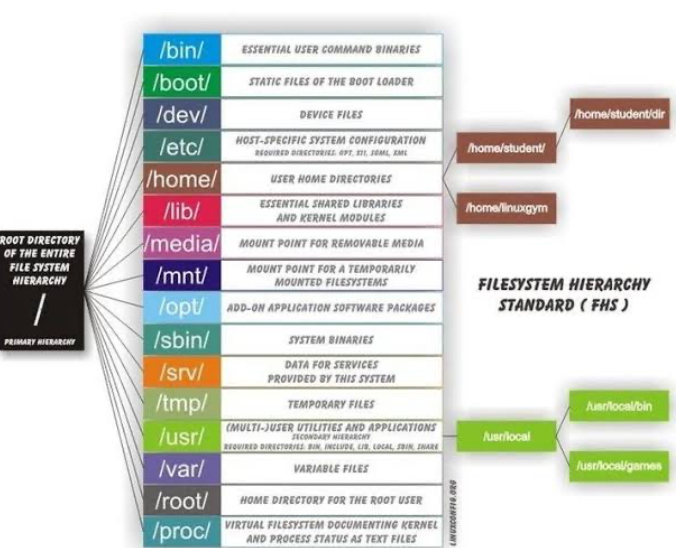
* **/bin**: Essential command binaries. Contains fundamental command-line utilities like ls, cp, and mv.
* **/boot**: Contains boot loader files and the Linux kernel. This is crucial for the system startup process.
* **/dev**: Device files that represent hardware components. For example, /dev/sda refers to the first hard disk.
* **/etc**: Configuration files for system and application settings. This includes user account information, network configurations, and more.
* **/home**: User home directories. Each user has a subdirectory here, such as /home/username.
* **/lib**: Essential shared libraries and kernel modules required by system binaries.
* **/media**: Mount points for removable media like USB drives and CDs.
* **/mnt**: Temporary mount points for mounting file systems.
* **/opt**: Optional application software packages.
* **/srv**: Data for services provided by the system, such as web or FTP servers.
* **/tmp**: Temporary files created by users and applications. This directory is often cleared on reboot.
* **/usr**: User-related programs and data. It contains subdirectories like /usr/bin for user commands and /usr/lib for libraries.
* **/var**: Variable files like logs and databases that change in size and content.

Figure 10Linux file system

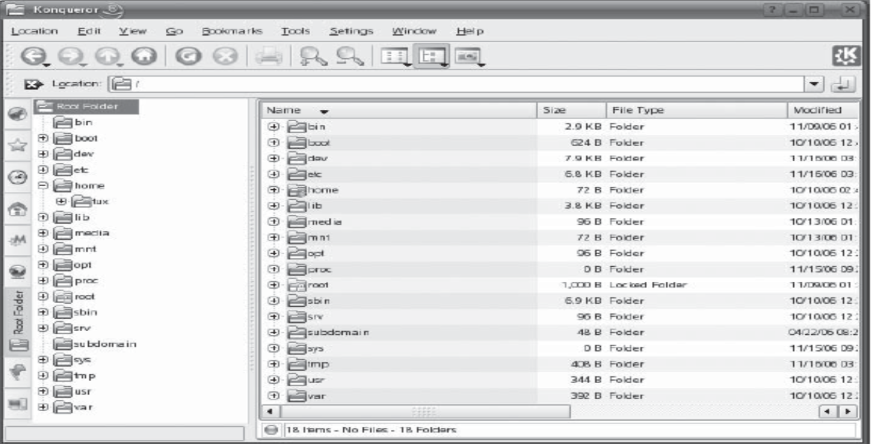


Figure 11linux file system (b)

**File Types**

Linux supports several types of files, including:

* **Regular files**: Contains user data.
* **Directories**: Special files that contain lists of other files and directories.
* **Special files**: Includes device files and named pipes.
* **Symbolic links**: Pointers to other files or directories.

**File Permissions**

Linux implements a robust permission system to enhance security. Each file and directory has associated permissions that dictate access levels:

* **Owner**: The user who owns the file.
* **Group**: A set of users that have shared access to the file.
* **Others**: All other users on the system.

**Permission Types**

* **Read (r)**: Permission to read the file.
* **Write (w)**: Permission to modify the file.
* **Execute (x)**: Permission to execute a file (for scripts and programs).

Permissions are represented in a three-character string format, such as rwxr-xr--, where the first three characters represent owner permissions, the next three represent group permissions, and the last three represent permissions for others.

**Mounting File Systems**

Linux allows multiple file systems to be mounted at different points in the directory tree. The mount command is used to attach file systems. For example:

Bash

**Common File System Types**

Linux supports various file system types, including:

* **ext4**: The most common file system used in modern Linux distributions, known for its performance and reliability.
* **XFS**: A high-performance file system suitable for large-scale data storage.
* **Btrfs**: A newer file system designed for advanced features like snapshotting and dynamic resizing.
* **FAT32/exFAT**: Used primarily for removable media due to broad compatibility.

**Study of linux editors with their command specifications**

Linux text editor is kind of a computer program that can edit text like notepad software. Text editors provide services in software development and operating system software which can be utilized to include programming languages, source code, documentation files etc.

Linux text editors can be used for editing text files, writing codes and updating user instruction files and many more.

Linux Text editors come in two forms-

**(i) Command Line Text editors** (such as Vi, nano, pico and more) and

**(ii) GUI based text editors** (such as GNOME, KDE, etc)

* **Introduction of Vi editor :-**

The default editor that comes with a UNIX os is Vi editor(Visual editor).

The unix editor is a full screen editor used to perform a variety of tasks. The unix architecture comes in three modes:

**(i) Command**, **(ii) insert** and **(iii) Esc mode**

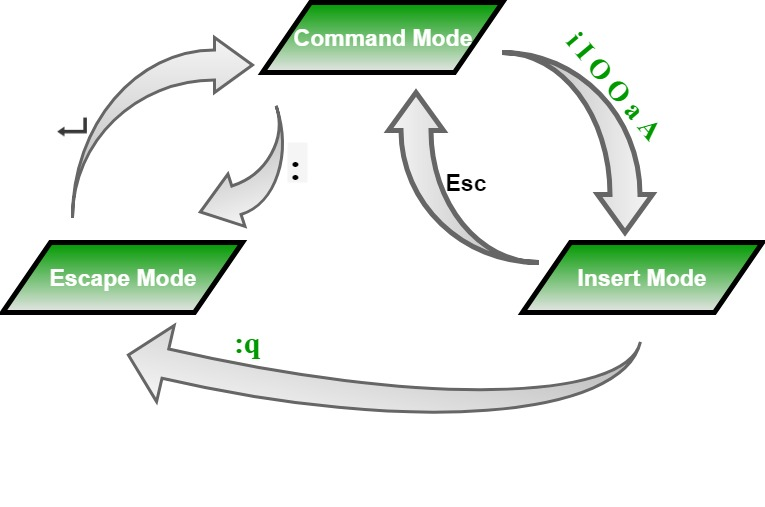
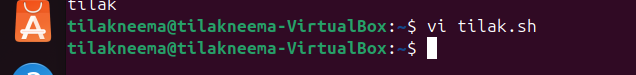
****

Figure 12 Modes of operation and their transition in VI editor

**1. Vi/Vim Editor**

Vim is a highly configurable text editor built to enable efficient text editing. It has a steeper learning curve compared to nano but offers powerful features.

* **Command to Open File**: vi filename or vim filename
* **Modes**: Command mode (for commands) and Insert mode (for text editing)
* **Commands**:
  + **i**: Enter Insert mode
  + **:w**: Save changes
  + **:q**: Quit
  + **:wq**: Save and quit
  + **:q!**: Quit without saving

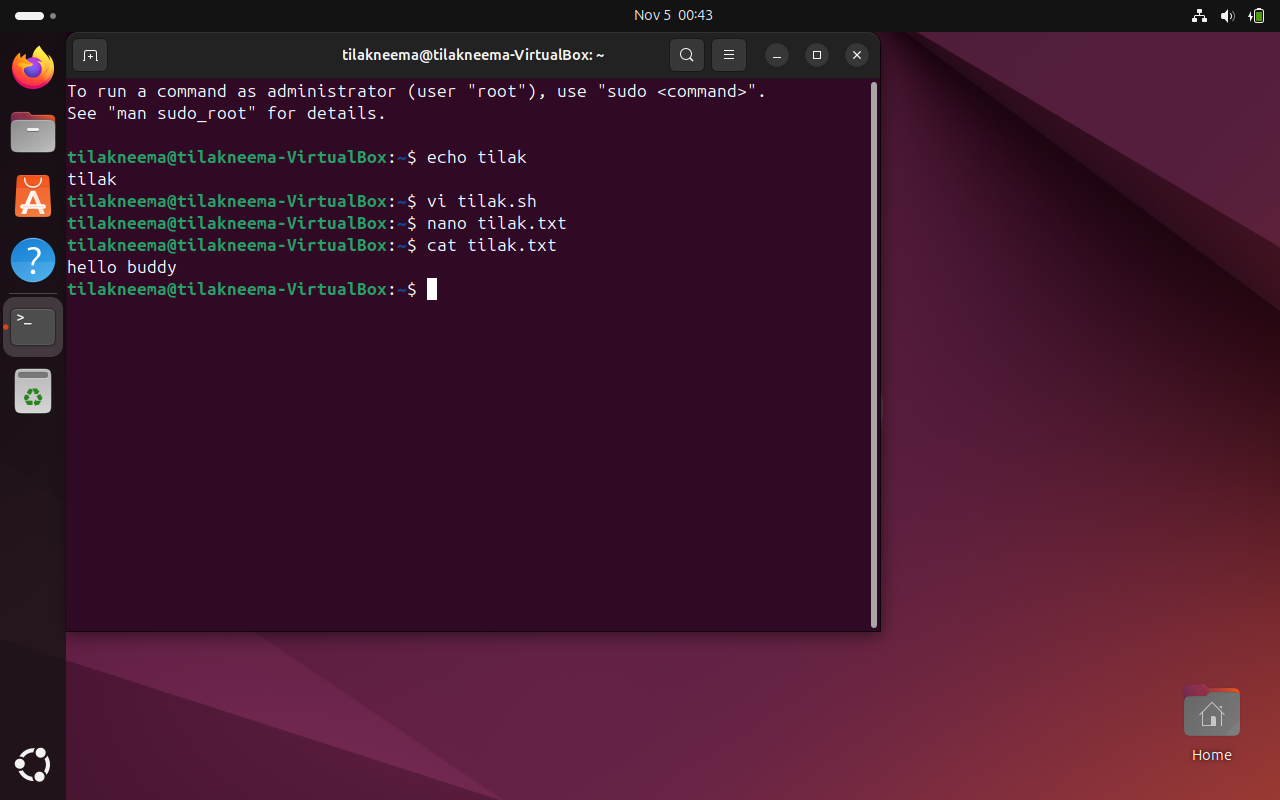


**Features:**

* Multiple modes (normal, insert, visual).
* Extensive plugins and configurations.
* Powerful search and replace capabilities.

**2. Nano Editor**

* **Command to Open File**: nano filename
* **Commands**:
  + CTRL + O: Save file
  + CTRL + X: Exit editor
  + CTRL + K: Cut current line
  + CTRL + U: Paste cut text

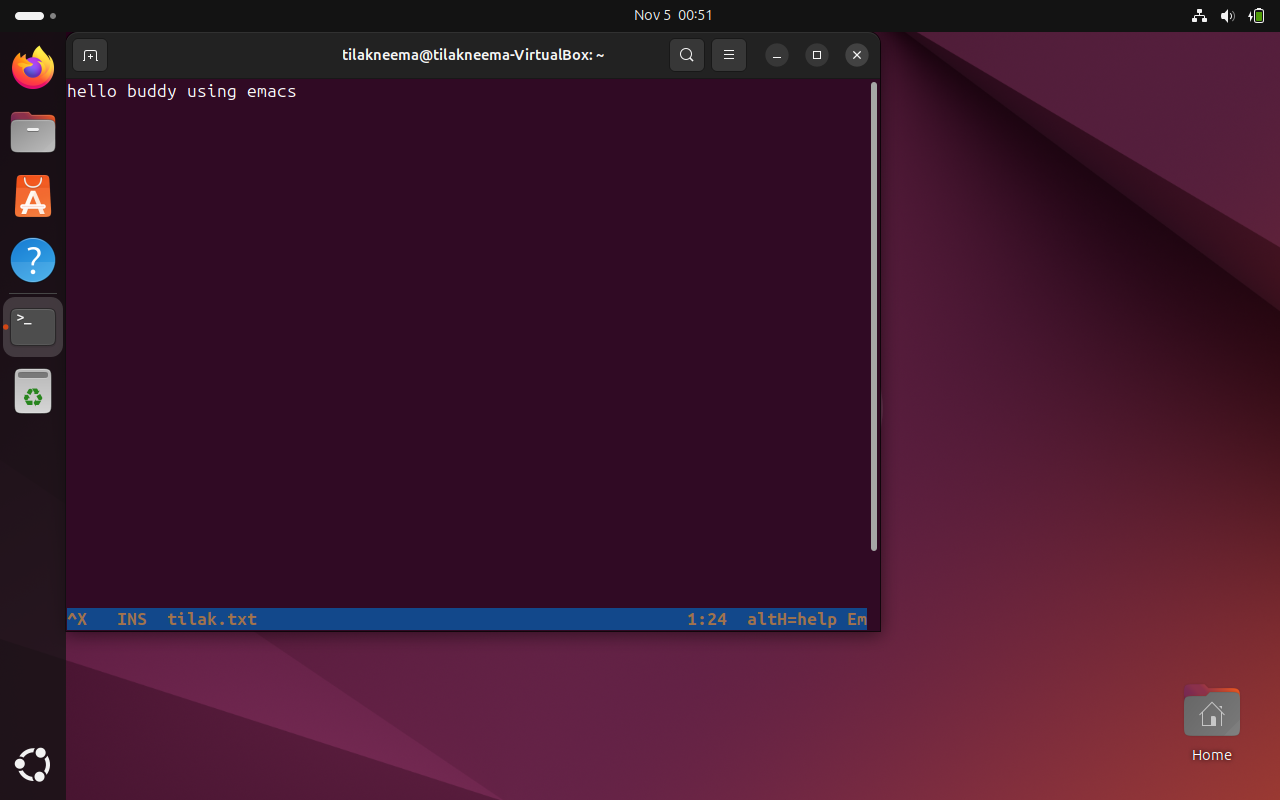


**Features:**

* Simple interface.
* On-screen command shortcuts.
* Supports syntax highlighting.

**3. Emacs Editor**

* **Command to Open File**: emacs filename
* **Commands**:
  + CTRL + X CTRL + S: Save file
  + CTRL + X CTRL + C: Exit editor
  + CTRL + \_: Undo last action
  + CTRL + K: Cut text from cursor to end of line



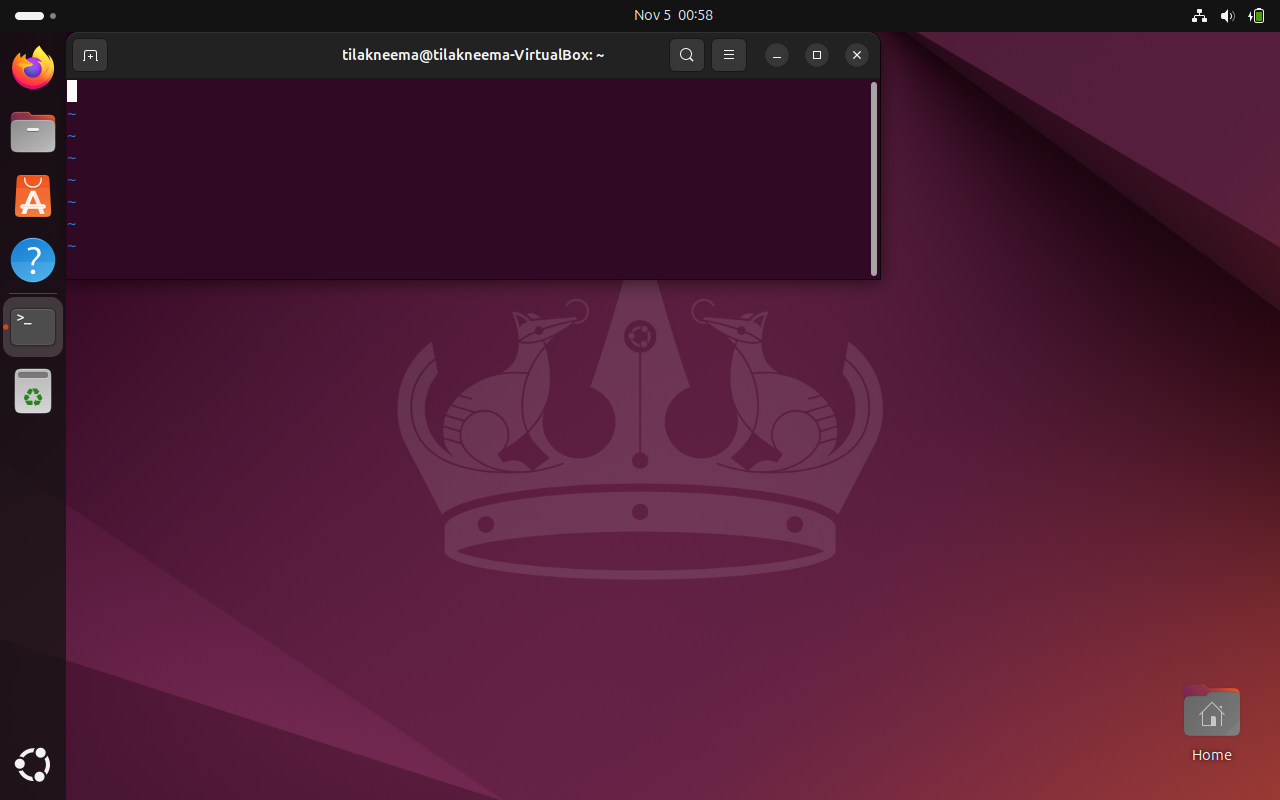
|  |  |  |  |
| --- | --- | --- | --- |
| **Command** | **Nano** | **Vim** | **Emacs** |
| Open a file | Nano filename | Vim filename | emacs filename |
| Save changes | ctrl+o | :w | Ctrl+x, ctrl+s |
| Exit | ctrl+x | :q | Ctrl+x, ctrl+c |
| Cut Text | ctrl+k | dd | ctrl+k |
| Paste Text | ctrl+u | p | ctrl+y |
| Search Text | ctrl+w | / | ctrl+s |
| Undo Changes | Not available | u | ctrl+\_ |

**Experiment-6**

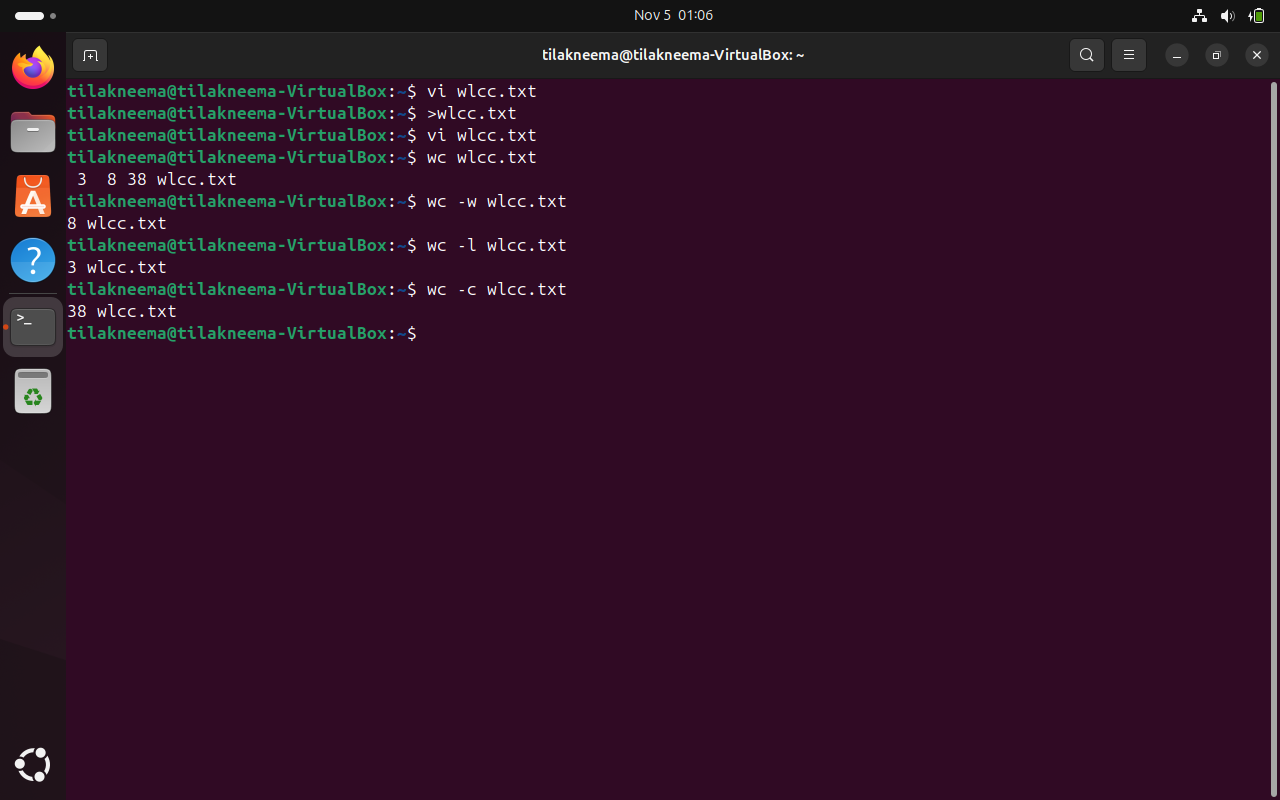
**Create a file called WLCC.txt with some lines and display how many lines, words and characters are present.**

To create a file named wlcc.txt and count lines, words, and characters:

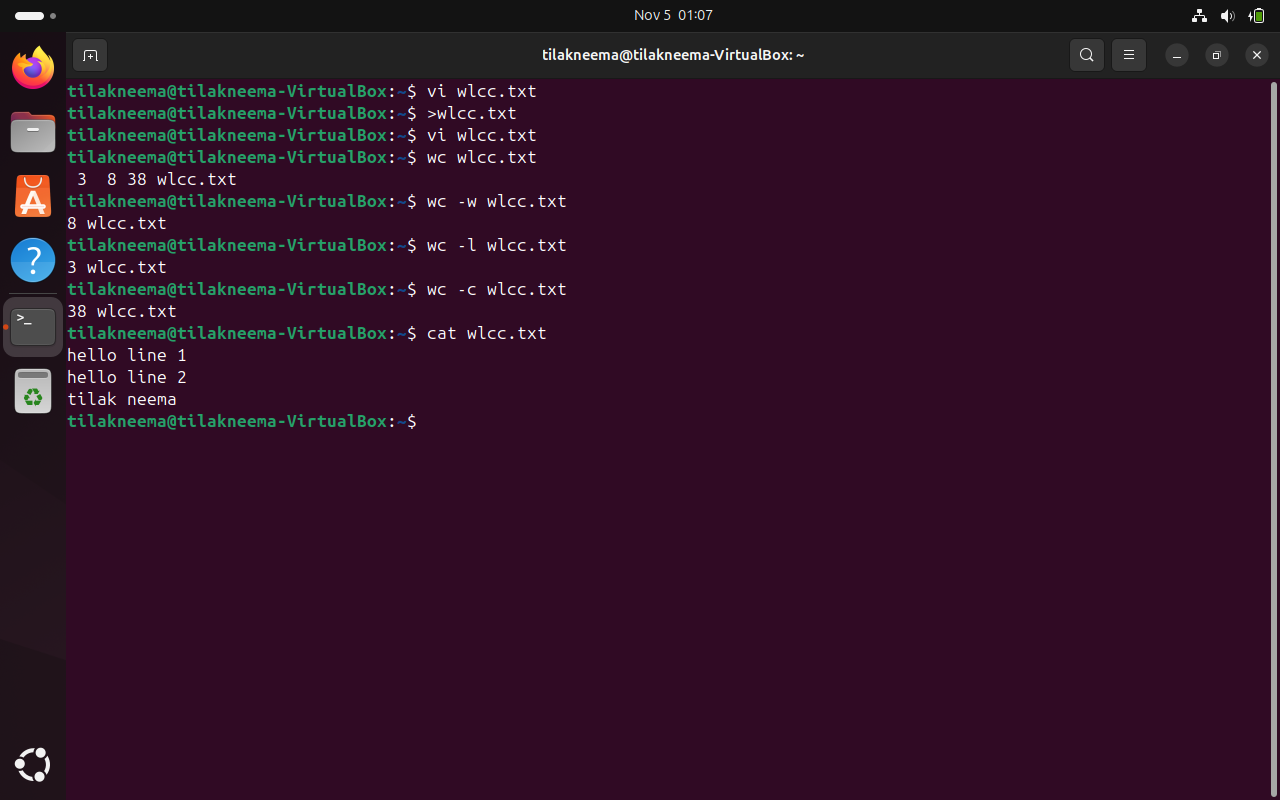
**Step-1**: vi command opens the text editor



**Step-2**: wc command is used to count the number of lines, words, and characters respectively.



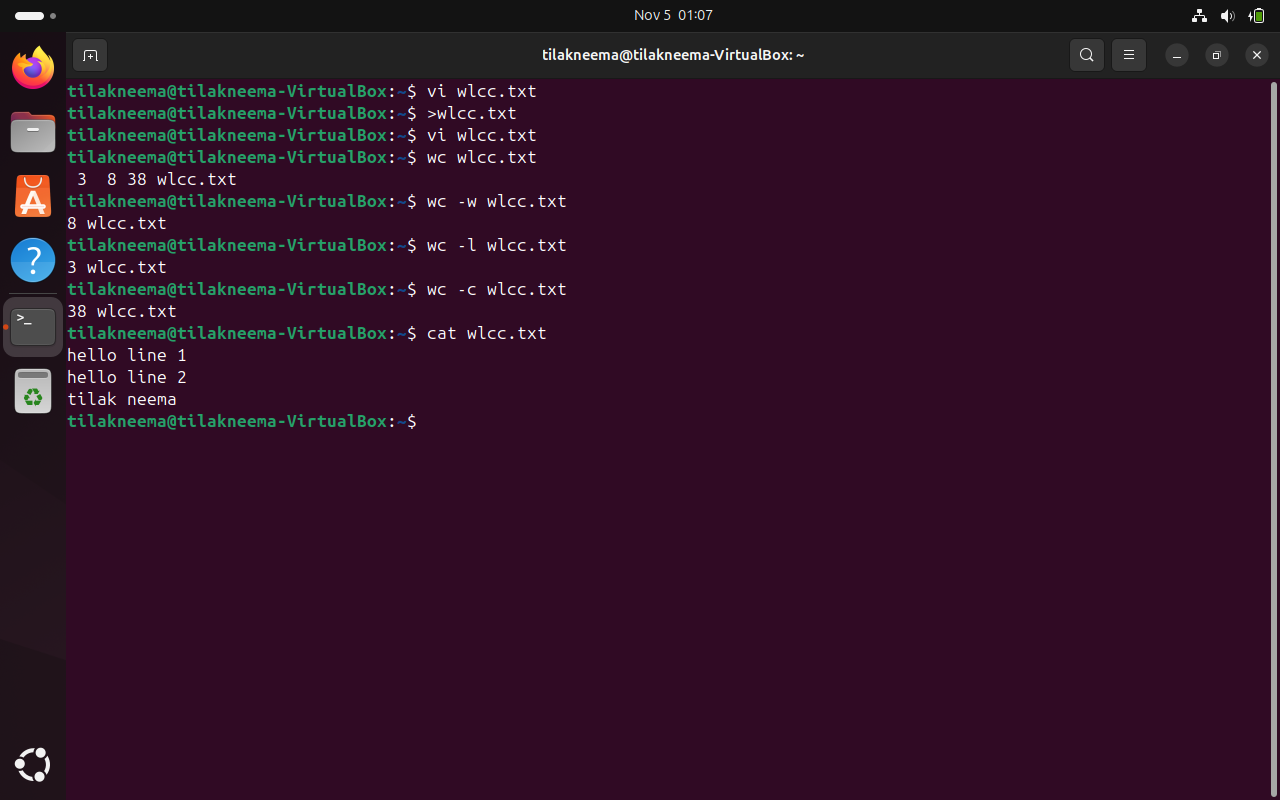
**Contents of wlcc.txt –**



**Experiment-7**

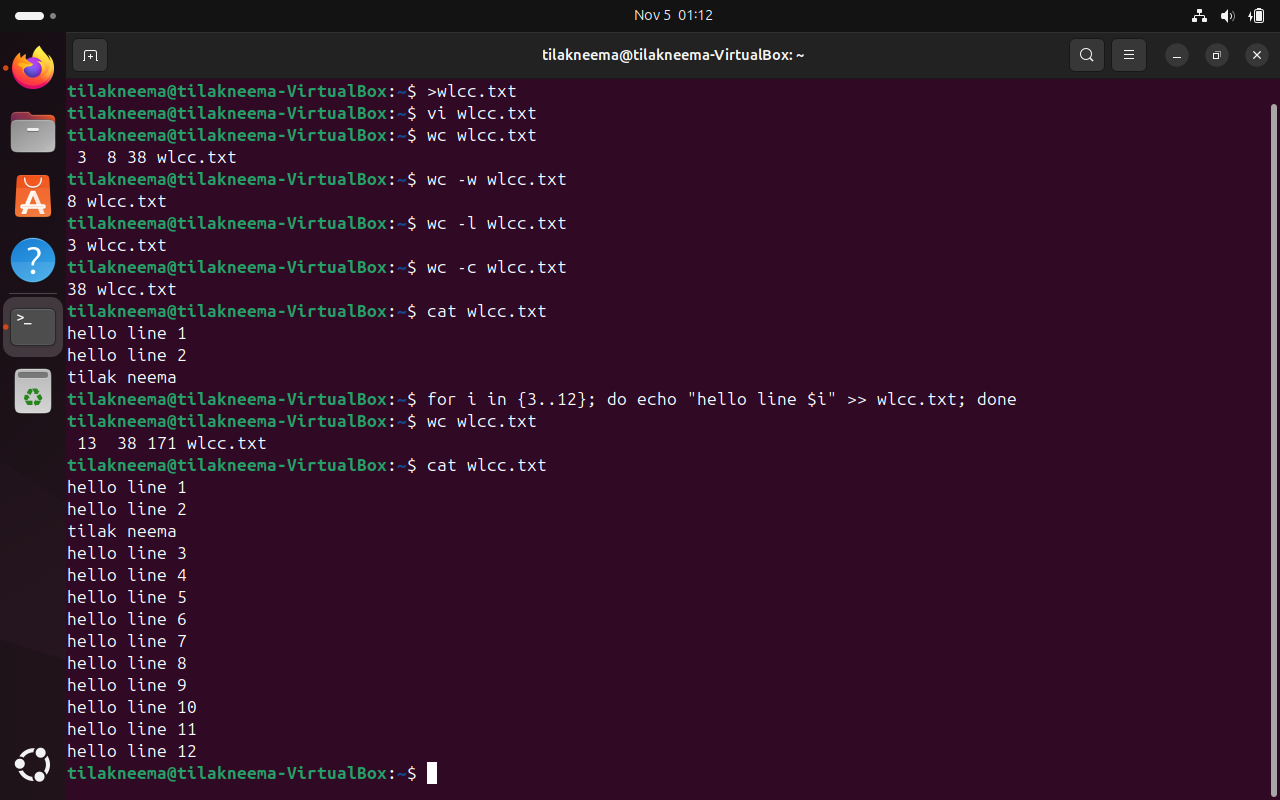
**Append 10 more lines in the file WLCC.txt and display**

**Contents of wlcc.txt –**



**Steps involved :-**

1. Opened the file wlcc.txt in vi editor.
2. Then pressed G followed by o to start editing the file from 5th line onwards.
3. After adding those lines pressed ESC and typed :wq to save the state of the file and quit.



**Experiment - 8**

**Study and use of commands for changing file permissions**

Linux uses permissions to control access to files and directories, which can be modified using the chmod command.

1. **Basic Permission Types**:
   * **r**: Read
   * **w**: Write
   * **x**: Execute
2. **Common chmod Commands**:
   * **Change Permissions by User Type**:
     + chmod u+x filename: Adds execute permission for the user (owner)
     + chmod g-w filename: Removes write permission for the group
     + chmod o+r filename: Adds read permission for others
   * **Using Numeric Mode**:
     + chmod 755 filename: Sets permissions to rwxr-xr-x
     + chmod 644 filename: Sets permissions to rw-r--r--

The basic syntax for **chmod** is

**chmod [options] [permissions] [file or directory]**

**Operations**:

* **+** : Adds the specified permission.
* **-** : Removes the specified permission.
* = : Sets the exact permission (replacing previous permissions).

To view the current permissions of a file or directory, we use **ls -l** command.

