History:

Linux was created by Linus Torvalds in 1991 as a free and open-source alternative to Unix.

It has since become a prominent operating system kernel and is widely used in server environments, embedded systems, and even desktops.

Foundation:

It is a non-profit organization that supports the growth of Linux and promotes collaboration in open-source projects.

It hosts various projects, including the Linux kernel, and provides resources, training, and events to the Linux community.

Hardware Requirements:

It is versatile and can run on a wide range of hardware, from low-power embedded devices to high-performance servers.

Minimal requirements typically include a CPU, RAM, storage, and compatible peripherals.

Components:

Kernel:

The core component, managing hardware resources, and providing essential services.

Shell:

The user interface to interact with the operating system through commands.

Utilities:

Various command-line and graphical tools for system management.

System Libraries:

Libraries providing essential functions to programs.

System Initialization:

The process that starts the system and initializes components.

Distributions:

Linux distributions or distros are variations of the Linux OS that bundle the Linux kernel with additional software and a package management system.

Examples: Ubuntu, Fedora, Debian, CentOS, and Arch Linux.

Features:

Open Source:

The source code is freely available, and users can modify and distribute their versions.

Multiuser and Multitasking:

Supports multiple users and concurrent execution of processes.

Security:

Linux has a robust security model with user permissions, firewalls, and encryption.

Stability and Reliability:

Known for its stability, even in high-load server environments.

Networking:

Powerful networking capabilities, making it a preferred choice for servers.

Choosing a Suitable Linux Distribution:

Consider the purpose (server, desktop, embedded).

Evaluate package management systems (e.g., apt, yum).

Assess community support and documentation.

Examine system requirements.

Consider user interface preferences (command-line vs. graphical).

Architecture

Kernel:

It is the core part of the OS, managing hardware resources, scheduling tasks, and providing essential services.

Modules can be dynamically loaded and unloaded, enhancing flexibility.

Architectural Differences between Windows and Linux

Monolithic Kernel:

Linux uses a monolithic kernel, where core functions are part of a single executable.

Microkernel:

Windows follows a more modular approach with a microkernel, separating core functions into different processes.

User Access Control (UAC):

Windows often uses UAC to manage administrative permissions, while Linux relies on user permissions and sudo.

Configuration & Customizations of Linux

Structure:

Filesystem Hierarchy Standard (FHS) defines the directory structure.

Key directories include `/bin` (executables), `/etc` (configuration files), `/home` (user home directories), and `/var` (variable data).

Installation:

Installation methods vary by distribution (graphical installer, text-based installer, network installation).

Partitioning, filesystem selection, and package selection are common installation steps.

Grub or other bootloaders are used to manage the boot process.

Different Ways to Install Linux

1. **Graphical Installer:**

 Commonly used in desktop-oriented distributions.

Provides a user-friendly interface guiding users through the installation process.

2. **Text-Based Installer:**

Suitable for server installations or systems with minimal resources.

Often used in server-oriented distributions.

3. **Network Installation:**

Installs the base system over the network, fetching packages as needed.

Useful for minimal installations or when an internet connection is available.

4. **Custom Installations:**

Allows users to customize package selection, partitioning, and system configuration.

Common in distributions targeting advanced users.

Linux Installation (CentOS 7 - Recommended)

1. **Download CentOS ISO:**
   Obtain the CentOS 7 ISO file from the official website.

2. **Create Bootable Media:**
   Burn the ISO to a DVD or create a bootable USB using tools like Rufus or dd.

3. **Boot from Installation Media:**
Start the computer from the bootable media.

4. **Start Installation:**
Follow on-screen instructions, selecting language, time zone, and keyboard layout.

5. **Partitioning:**
Choose partitioning options, such as automatic or manual partitioning.

Assign mount points and file systems to partitions.

6. **Software Selection:**
Select software packages to install, depending on the system's purpose (minimal, server, desktop).

7. **Configure Network:**
Set up network settings, including IP address, hostname, and DNS.

8. **Create User Accounts:**

Set root password and create user accounts.

9. **Complete Installation:**

Allow the installer to complete the installation process.

CentOS vs. CentOS Stream

CentOS:

Traditional stable release with long-term support.

Updates focus on stability and security.

Ideal for production servers.

CentOS Stream:

Rolling release model with more frequent updates.

Provides a preview of upcoming RHEL (Red Hat Enterprise Linux) features.

Suitable for those who want the latest features and can adapt to changes more quickly.

Take a Snapshot of VM

In virtualization environments (e.g., VMware, VirtualBox):

VM Snapshot:

Captures the current state of a virtual machine.

Useful before major changes or updates.

Facilitates easy rollback in case of issues.

Boot Process

1. **BIOS/UEFI:**

   Initializes hardware and loads the bootloader.

2. **Bootloader (GRUB):**

Presents a menu to choose the operating system.

Loads the kernel into memory.

3. **Kernel:**

Initializes hardware, mounts the root filesystem, and starts the init process.

4. **Init Process:**

Invokes system initialization scripts.

Transitions to the default runlevel or target.

5. **Runlevel / Target:**

Defines the system state (e.g., single-user mode, multi-user mode).

Executes startup scripts accordingly.

Partitioning

Disk Partitioning (fdisk):

Use the `fdisk` command to create, delete, and modify partitions.

Example: `fdisk /dev/sda`

Check Disk Space (df):

The `df` command displays disk space usage.

Example: `df -h`

Dual Boot

Dual Boot Setup:

Install multiple operating systems on the same computer.

Requires partitioning and a bootloader (e.g., GRUB) to manage OS selection during boot.

Virtual Memory and Swap Space

Swap Space:

A dedicated space on the disk used as virtual memory.

Improves system performance by providing additional memory when physical RAM is full.

Create and activate swap space using the `mkswap` and `swapon` commands.

Adding Swap Space:

Create a swap partition or use a swap file.

Example (creating a 1GB swap file):

```
dd if=/dev/zero of=/swapfile bs=1M count=1024
mkswap /swapfile
swapon /swapfile
```

File System & Storage Management

Structure:

Root Directory (/):

Top-level directory, contains all other directories and files.

Common Directories:

`/bin`: Binary executables.

`/etc`: Configuration files.

`/home`: User home directories.

`/var`: Variable data like logs and temporary files.

`/tmp`: Temporary files.

`/usr`: User programs and data.

`/lib`: Libraries.

Navigation Commands

cd (Change Directory):

`cd directory`:

Change to a specified directory.

`cd ..`:

Move to the parent directory.

`cd ~`:

Move to the home directory.

ls (List):

`ls`:

List files in the current directory.

`ls -l`:

Detailed list with permissions, owner, size, and modification date.

`ls -a`:

List including hidden files.

pwd (Print Working Directory):

`pwd`:

Display the full path of the current working directory.

Absolute and Relative Paths:

Absolute Path:

Specifies the complete path from the root directory.

Example: `/home/user/documents/file.txt`

Relative Path:

Specifies the path relative to the current directory.

Example: `../../folder/file.txt`

Creating Files and Directories:

touch:

`touch filename`: Create an empty file or update the timestamp of an existing file.

mkdir (Make Directory):

`mkdir directoryname`: Create a new directory.

cp (Copy):

`cp source destination`:

Copy files or directories.

`cp -r sourcedir destination`:

Recursively copy directories.

Disk Partitions and File Systems:

Partitioning:

Divide a physical disk into logical sections.

Common tools: `fdisk`, `parted`.

File Systems:

Defines how data is organized and stored on a storage device.

Common file systems: ext4, NTFS, FAT32.

Mounting and Unmounting File Systems:

mount:

`mount device directory`:

Mount a file system to a specified directory.

Example:

`mount /dev/sda1 /mnt`

umount:

`umount directory`:

Unmount a file system.

Example:

`umount /mnt`

Managing Disk Space (df, du):

df (Disk Free):

`df -h`:

Display disk space usage in a human-readable format.

`df -i`:

Display inode information.

du (Disk Usage):

`du -h`:

Display directory space usage in a human-readable format.

`du -s`:

Display total space used by a directory.

Working with Files & Directories:

1. **Regular Files:**

   Contain data (text, binary, etc.).

   Identified by `-` in the file permissions.

2. **Directories:**

   Containers for files and subdirectories.

   Identified by `d` in the file permissions.

3. **Symbolic Links (symlinks):**

   Pointers to another file or directory.

   Identified by `l` in the file permissions.

4. **Device Files:**

   Represent physical or virtual devices.

   Two types: Character (c) and Block (b) devices.

5. **FIFO (Named Pipes):**

Special files used for interprocess communication.

Identified by `p` in the file permissions.

6. **Sockets:**

Communication endpoints for processes.

Identified by `s` in the file permissions.

Commands for File and Directory Manipulation:

cp (Copy):

`cp source destination`:

Copy files or directories.

`cp -r sourcedir destination`:

Recursively copy directories.

rm (Remove):

`rm file`:

Remove (delete) a file.

`rm -r directory`:

Recursively remove a directory.

mv (Move):

`mv source destination`:

Move or rename files and directories.

mkdir (Make Directory):

`mkdir directoryname`:

Create a new directory.

rmdir (Remove Directory):

`rmdir directory`:

Remove an empty directory.

File Display Commands:

cat (Concatenate):

`cat filename`:

Display the entire content of a file.

less:

`less filename`:

View file content with navigation capabilities.

more:

`more filename`:

Display file content page by page.

head:

`head filename`:

Display the first few lines of a file.

tail:

`tail filename`:

Display the last few lines of a file.

Redirection:

Redirecting Output (`>` and `>>`):

 `command > file`:

Redirect command output to a file (overwrite).

 `command >> file`:

Redirect command output to a file (append).

Redirecting Input (`<`):

`command < file`:

Take input from a file for a command.

Piping (`|`):

 `command1 | command2`:

Redirect output from command1 as input to command2.

Files and Directory Permissions (chmod):

chmod (Change Mode):

Changes file permissions.

Syntax: `chmod [options] mode file`.

Example: `chmod +x script.sh` (add execute permission).

File Ownership Commands (chown, chgrp):

chown (Change Owner):

Changes the owner of a file or directory.

Syntax: `chown [options] owner:group file`.

Example: `chown user:group file.txt`.

chgrp (Change Group):

Changes the group of a file or directory.

Syntax: `chgrp [options] group file`.

Example: `chgrp staff file.txt`.

Changing Password:

passwd:

`passwd`: Change the user's password.

Users can change their own passwords or the root user can change any password.

Notes:

Understanding and managing file types is crucial for effective system administration.

Commands like `find` and `locate` can be used for searching files and directories.

Proper handling of file permissions and ownership is essential for security.

Redirection and piping are powerful concepts for managing command input and output.

Editors:

vi (Visual Editor):

Classic text editor in Linux.

Modes:

Command Mode: For navigation and other commands.

Insert Mode: For actual text entry.

Visual Mode: For text selection.

Difference between vi and vim Editors:

vim (Vi Improved):

An extended and improved version of vi.

Enhanced features include syntax highlighting, multiple undo/redo, and more.

nano and Pico:

nano:

User-friendly text editor with easy-to-understand shortcuts.

Good for beginners.

pico:

Text editor similar to nano.

Part of the Pine email client.

sed Command:

sed (Stream Editor):

Used for text stream processing.

Enables text manipulation using commands specified in a script.

Filters / Text Processing Commands

cut:

Extracts specific columns from a text file.

awk:

Pattern scanning and processing language.

Useful for data extraction and reporting.

grep/ egrep

grep (Global Regular Expression Print):

Searches for patterns in a text.

egrep (Extended grep):

Supports extended regular expressions.

sort/uniq

sort:

Sorts lines of text.

uniq:

Removes duplicate lines from a sorted file.

wc (Word Count):

Counts lines, words, and characters in a file.

compare files (diff and cmp)

diff:

Compares two files and shows the differences.

cmp:

Compares two files byte by byte.

Compress and Uncompress (tar, gzip, gunzip)

tar:

Creates and manipulates archive files.

gzip/gunzip:

Compresses and decompresses files.

User Account Management:

useradd:

Adds a new user to the system.

groupadd:

Creates a new user group.

usermod:

Modifies user attributes.

userdel:

Deletes a user account.

groupdel:

Deletes a user group.

Switch Users and sudo Access (su, sudo)

su (Switch User):

Switches to another user.

sudo (Superuser Do):

Allows a permitted user to execute a command as the superuser or another user.

Monitor Users (who, last, w, id):

who:

Displays information about currently logged-in users.

last:

Shows a list of last logged-in users.

w:

Displays information about currently logged-in users and their activities.

id:

Displays user and group information.

System Utility Commands:

Date and Time:

date:

Displays the current date and time.

Syntax: `date [options]`.

System Uptime:

uptime:

Shows how long the system has been running.

Displays system load averages.

Host Information:

hostname:

Displays the system's hostname.

Can be used to set the hostname.

System Information:

uname:

Prints system information.

Commonly used with options like `-a` (all information).

Finding Executables:

which:

Locates the executable of a command in the system's PATH.

Example: `which ls` will show the path to the `ls` command.

Calendar:

cal:

Displays a calendar.

Options include `-3` (show previous, current, and next month) and `-y` (show the entire year).

Calculator:

bc (Basic Calculator):

Command-line calculator supporting basic arithmetic.

Interactive mode or scriptable with files.

Process Management & System Monitoring:

Process Status:

ps:

Displays information about active processes.

Options like `aux` show detailed information.

Background and Foreground Jobs:

bg:

Puts a job in the background.

Example: `bg %1` (puts job 1 in the background).

fg:

Brings a job to the foreground.

Example: `fg %1` (brings job 1 to the foreground).

Process Priority:

nice:

Adjusts the priority of a process.

Example: `nice -n 10 command` (runs 'command' with a lower priority).

Troubleshooting:

Network Configuration:

ifconfig:

Displays and configures network interfaces.

Example: `ifconfig eth0` shows information about the eth0 interface.

Network Testing:

ping:

Tests network connectivity by sending ICMP echo requests.

Example: `ping google.com`.

traceroute:

Shows the route packets take to reach a destination.

Example: `traceroute google.com`.

DNS Troubleshooting:

nslookup:

Queries DNS servers for domain information.

Example: `nslookup google.com`.

dig (Domain Information Groper):

A versatile tool for DNS troubleshooting.

Example: `dig google.com`.

host:

Performs DNS lookups.

Example: `host google.com`.

Web and Database Services:

Configuring Web Servers:

Apache:

A widely used open-source web server.

Configuration file: `/etc/httpd/conf/httpd.conf`.

Commands: `service httpd start/stop/restart`.

Nginx:

A lightweight and high-performance web server.

Configuration file: `/etc/nginx/nginx.conf`.

Commands: `service nginx start/stop/restart`.

Setting up Databases:

MySQL:

A popular relational database management system.

Configuration file: `/etc/my.cnf`.

Commands: `service mysqld start/stop/restart`.

PostgreSQL:

An open-source relational database system.

Configuration file: `/etc/postgresql/VERSION/main/postgresql.conf`.

Commands: `service postgresql start/stop/restart`.

Managing Web Applications:

Web applications often use server-side technologies like PHP, Python, or Node.js.

Common directories: `/var/www/html` for Apache, `/usr/share/nginx/html` for Nginx.

Use of frameworks and content management systems (e.g., WordPress, Drupal).

Automation and Scripting:

Introduction to Shell Scripting (Bash):

Bash is a popular shell in Linux.

Scripting involves writing a series of commands to automate tasks.

Automating Tasks with Scripts:

Writing scripts to automate repetitive tasks.

Scheduling scripts using tools like cron.

Configuration Management Tools (Ansible):

Ansible automates software provisioning, configuration management, and application deployment.

Uses YAML for configuration files.

Shell Scripting:

Types of Shells:

Common shells include Bash, sh, zsh, and csh.

Bash is widely used and the default shell in many Linux distributions.

Starting a Shell:

Open a terminal to access the shell.

echo $SHELL` to check the current shell.

Creating Your First Script - Hello World:

Example:

```
 #!/bin/bash
 echo "Hello, World!"
```

Conditions/If-Else Statements:

Example:

```
#!/bin/bash
age=18
if [ "$age" -ge 18 ]; then
    echo "You are an adult."
else
    echo "You are a minor."
fi
```

Case Statements Script:

Example:

```
#!/bin/bash
fruit="apple"
case $fruit in
    "apple")
        echo "It's an apple.";;
    "banana")
        echo "It's a banana.";;
    *)
        echo "Unknown fruit.";;
esac
```

For Loop Script, Do-While Scripts:

Example:

```
#!/bin/bash
for i in {1..5}; do
    echo "Number: $i"
done
```

Do-While Loop Example:

```
#!/bin/bash
count=1
while [ $count -le 5 ]; do
    echo "Count: $count"
    ((count++))
done
```

Exit Status:

Example:

```
#!/bin/bash
command
if [ $? -eq 0 ]; then
    echo "Command succeeded."
else
    echo "Command failed."
fi
```

Introduction to GCC Compiler:

Basics of GCC:

GCC (GNU Compiler Collection):

A compiler system for various programming languages.

Commonly used for compiling C and C++ programs.

Compilation of Program:

Command: `gcc program.c -o program`

Execution of Program:

Command: ./program

Time Stamping:

Command:

```
  gcc -o program program.c -D__DATE__="$(date)" -D__TIME__="$(date
+"%T")"
```