Linux Programming: Assignment-1

Samriddhi Guha | ENG24CY0157 | 3-C (25)

1. What is Linux Operating System (OS)? List three pros and cons of it. (CO1)

Linux is a free, open source, Unix - like operating system created by Linus Torvalds in 1991, that helps interaction between a computer's hardware and software. There are several distributions that combine the Linux kernel (the central, core component of the operating system that acts as the primary interface between the software and the hardware) with other software such as Ubuntu, Kali Linux, ParrotOS, etc.

Pros:

- Open Source: Programmers have access to the source code which they can choose to view, modify or share as per the use case.
- Secure and Reliable: Linux distributions are known to be secure due to its user permission model, which requires explicit permission for administrative tasks using the "sudo" keyword.
- Customisable: As it is open source, it gives users the ability to customise the operating system as per their wish.

Cons:

- Learning Curve: Users migrating from other operating systems such as Windows or macOS may find it challenging to interact with the command line directly without any graphical user interface.
- Gaming: Linux distributions are notoriously known for their compatibility issues with gaming experiences, which are significantly being improved.
- The Dilemma Of Choice: The variety and essence of distributions using the Linux kernel may make it hard for beginners to learn and use a particular one.

2. Differentiate between Linux, Mac, Android, and Windows OS with at least six unique features. (CO1)

Feature	Linux	macOS	Android	Windows
Core Architecture & Kernel	Unix-like system built on the **Linux kernel.**	Built on the **XNU kernel** that combines Mach microkernel and FreeBSD components.	Modified version of the **Linux kernel** built specifically for mobile phone use cases.	Built on the hybrid **Windows NT kernel,** which is not Unix-like.

Feature	Linux	macOS	Android	Windows
Primary User Interface	Highly customizable. Users can choose from numerous **Desktop Environments** like GNOME, KDE, or XFCE, or use a **command-line interface.**	**Aqua GUI**, known for its Dock and Menu Bar. The underlying UI is Core Animation. Tightly integrated and consistent.	**Material You** is a touch-centric graphical interface with customizable widgets and an app drawer.	**Fluent Design System** (formerly Metro). Centered around the Start Menu and Taskbar for a desktop-oriented experience.
Software Distribution	Primarily through centralized **package managers** (e.g., APT for Debian/Ubuntu, YUM/DNF for Fedora/RHEL). Source code is readily available.	**Mac App Store** for curated apps and direct downloads of .dmg or .pkg files from developers. Includes the Homebrew package manager.	Primarily through the **Google Play Store**. Sideloading of .apk files is possible but requires user permission.	**Microsoft Store** and standalone executable installers(.exe,.msi) . Includes package managers like Chocolatey and Winget.
Hardware Integration	Runs on the **widest range of hardware**, from servers and desktops to embedded systems and supercomputers.	Runs exclusively on **Apple hardware** (MacBooks) which allows for high optimization of performance and battery life.	Runs on a **vast array of third-party hardware** from various manufacturers (Samsung, Google, etc.), leading to a fragmented ecosystem.	Runs on a **wide variety of hardware** from numerous manufacturers (Dell, HP, Lenovo, etc.), making it the default OS for most pre-built PCs.

Feature	Linux	macOS	Android	Windows
Licensing Model & Cost	The kernel and most software are **Free and Open Source Software (FOSS)** under the GNU General Public License. It is free of charge.	**Proprietary**. The OS is included with the purchase of Apple hardware and is not licensed for use on other systems.	**Open Source** (Android Open Source Project - AOSP), but the Google Mobile Services (GMS) layer (Play Store, Gmail) is proprietary.	**Proprietary**. Requires the purchase of a license, which is typically included in the price of a new computer.
Primary Target Audience	**Developers, system administrators, and technical users** who value customization, control, and open-source principles.	**Creative professionals, students, and general consumers** who prioritize ease of use, a stable user experience, and a strong ecosystem.	**Mobile device users**. It is the dominant OS for smartphones and tablets, focusing on touch input and mobile connectivity.	**General desktop and business users**. It has the largest market share for desktop computers and extensive enterprise support.

3. Why is Linux preferred for Mainframe Servers for legacy applications? Give three out-of-the-box technical reasons. (CO1)

Some of Linux’s most appreciable features include cost effectiveness and flexibility to modernise and integrate older systems without sacrificing the mainframe’s core strengths, that makes it a preferable choice for Mainframe Servers in running legacy applications.

- Linux can run thousands of isolated virtual servers on one mainframe, allowing secure and high speed communication between old and new applications without compromising on smoothness and efficiency.
- It uses the mainframe’s dedicated input and output processors to handle tasks involving network and data, protecting the performance of core legacy systems.
- Linux allows programmers to use tools like Python directly on the mainframe, making it faster and more secure to build modern interfaces for old applications.

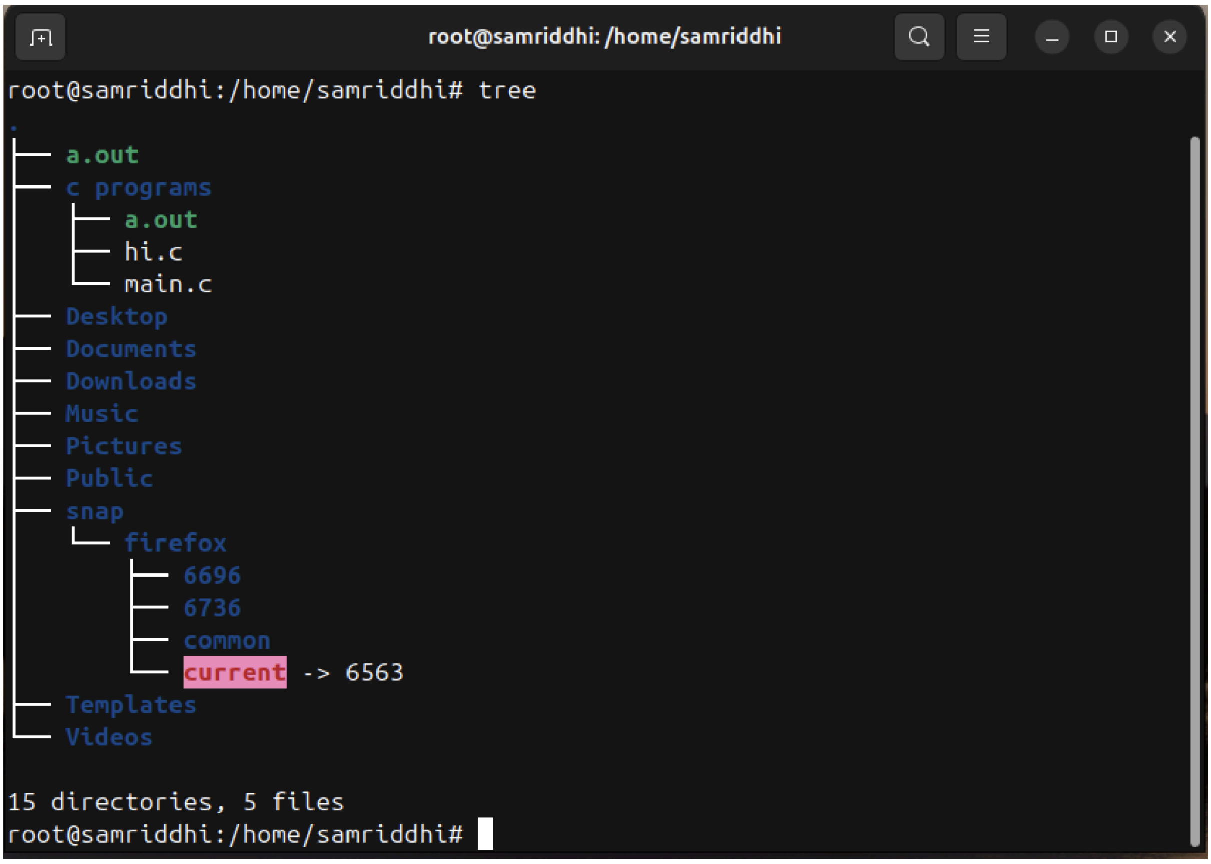
4. Explain the structure of the Linux File System with a proper diagram. Note: you can use the tree command to find it out. (CO2)

The Linux file system is a tree-like hierarchy of directories that start from a single root directory, analogous to a real life tree with its roots beneath the earth. Everything in Linux, including hardware devices are treated as files and exist under this single tree.

On the CLI of any Linux distro, one can install and witness the use of tree command using the following command:

sudo su
sudo apt install tree
root@samriddhi:/home/samriddhi# tree

Executing the commands above displays the output shown



```
root@samriddhi:/home/samriddhi# tree
.
├── a.out
├── c programs
│   ├── a.out
│   ├── hi.c
│   └── main.c
├── Desktop
├── Documents
├── Downloads
├── Music
├── Pictures
├── Public
├── snap
│   └── firefox
│       ├── 6696
│       ├── 6736
│       ├── common
│       └── current -> 6563
├── Templates
└── Videos

15 directories, 5 files
root@samriddhi:/home/samriddhi#
```

5. If Linux OS is open-source, how do companies like Red Hat still making money from it? Do a market study and answer properly. (CO2)

While Linux is well known for being open source and free, companies like Red Hat do not directly make money from Linux itself, rather they sell subscriptions for their own version of Linux that is built on top of the base kernel. This may be particularly beneficial for certain businesses and organisations due to the following reasons:

- Security: Companies like Red Hat have dedicated security teams that find, patch and manage vulnerabilities (CVE's) flaws to provide an increased layer of security, so that businesses can trust them with critical data.
- Long-Term Stability: The company offers Long-Term Support (LTS) and guarantees stable updates for over 10 years so businesses relying on it can run smoothly for each version of RHEL (Red Hat Enterprise Linux). They provide security patches, bug fixes and support to their buyers.
- Customer Support: Red Hat subscription provides access to a global team of expert engineers who can rapidly troubleshoot and resolve complex issues, and who can be approached 24/7.
- Certification: Red Hat comes with a guarantee that their Linux will work with specific hardware and enterprise software.
- Curated Ecosystem: Red Hat uses RHEL as the foundation for a powerful ecosystem of enterprise products like OpenShift and Ansible. Getting companies to standardize on RHEL makes it much easier and more profitable and higher - level solutions.

6. Write the command to display today's date and time (i.e., current System time). (CO1)

The ***date*** command is used to display the date and time of the execution of the command.

```
samriddhi@samriddhi:~$ date
Tue Sep 23 03:56:40 PM IST 2025
```

7. Which command is used to check how long the system has been running? (CO1)

The ***uptime*** command is used to check how long the system has been running, the number of logged-in users and the system load averages for the last 1, 5, and 15 minutes. The execution of the code is as follows:

```
samriddhi@samriddhi:~$ uptime
15:59:21 up 5 min,  1 user,  load average: 0.04, 0.08, 0.03
```

8. What is the difference between shutdown -h now and halt? (CO1)

shutdown -h:

- The shutdown -h command provides a more orderly and controlled way of shutting down the system. It does so by allowing a delay or scheduled time for shutting down, and it can also send a message to all logged in users before it begins.
- Normally shutdown sends a power off sequence after bringing the system down, effectively turning off the hardware. The -h option identifies stopping and powering off.
- It is considered more orderly as it allows users to save work and ensures all services have properly stopped.

halt:

- The halt command ends every process, syncs the filesystem and stops the CPU state.
- Historically, halt was defined in a way that the hardware would remain on (which means the machine would not power off).
- In modern systems, safe halt can have similar behavior to poweroff or shutdown -h now, as the system can be set up to power off automatically after halt, but the main difference is that halt is directed at stopping the CPU and kernel processes.

Upon comparing them directly, we get the following differences:

- Power Management: halt in the past has typically left the system powered on, whereas shutdown (especially along with -h or in modern configurations) usually means powering down the hardware.
- Scheduling and User Engagement: shutdown provides options for a delayed shutdown, scheduled times, and broadcast notifications to users, while halt does not.
- Graceful Stop: shutdown is preferred generally because it provides a more orderly exit from the system, while halt is primarily an ungraceful method that focuses heavily on stopping system executions.

9. Compare init 0 and shutdown -h. Which is safer? Why? (CO1)

The command "shutdown -h" is safer. This is because it is designed to be a clean and friendly utility that permits time for services to exit gracefully, while init 0 is a lower-level command that is more abrupt.

The shutdown command is safer because it is designed for a multi-user environment. It provides an important grace period for both users and system services. By warning the user, it provides time to save their work and exit. By signaling the services to terminate gracefully, it avoids corruption of data and allows applications to be able to close their "files" and save their "states" properly. Init 0 is an immediate forceful command, without the grace period.

10. How do you check the manual page for any Linux commands? (CO1)

The manual page for any command can be checked using the *man* command. It formats and displays the online manual pages, which is built-in documentation for the system. man provides the most details of a command, including what options are available, its syntax, and what the command actually does.

Syntax: *man [command name]*
Example: *samriddhi@samriddhi:~$ man ls*

```
NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List  information  about  the FILEs (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort  is  specified.

…(continuation)
```

10) A system administrator accidentally powers off a Server machine without shutting it down properly. What problems can occur to the said Server?

- Unclean power off can cause filesystem issues where the journal must be replayed and in unlucky cases metadata or recent writes may be lost especially on poorly configured barriers or caches.

- Databases and message queues can require recovery or reindexing and RAID or storage stacks may resync while application state not flushed to disk can be corrupted or lost. Repeated sudden power losses can also damage disks increase SMART errors and escalate mean time to failure in storage hardware over time

---

Brainstorming:

a) As Linux Kernel is open-source, can we build our own operating system?

Yes. The main reason there are so many operating systems, commonly referred to as Linux distributions, is because the Linux kernel is open-source. An operating system is more than just the kernel. One would build all of the software components that constitute a functioning OS on top of the Linux kernel. A breakdown for the same is as follows.

- The Kernel: This is the core of the OS, and is where we would begin. The latest source code can be downloaded and configured to suit our needs.
- System Libraries & Utilities: A kernel by itself does not do anything. You need utilities to interface with it. Many come from the GNU Project (like bash and core utilities like ls, cp, mv). This is why it is often called "GNU/Linux".
- Other Software: Next you would add other components, such as a package manager (to install software), a graphical desktop (such as GNOME or KDE), and applications to round out our Linux system.
- Projects like Linux From Scratch (LFS) are famous for providing an instruction manual in order to guide the reader with step by step instructions to build a fully custom Linux system from the source code.

b) In order to do that, what are the stoppers, hurdles, and challenges?

Creating a custom operating system from the Linux kernel can be a significant undertaking that is frequently likened to constructing a car one nut and bolt at a time. The task is considerable.

The greatest challenge is the overwhelming amount of specialized knowledge you must possess. You need to have a solid understanding of:

- Low-level Programming: You need to know how to program in C and at least a little Assembly language, since that is what the kernel and core utilities are written in.

- Computer Architecture: You need to know how CPUs work, how memory (RAM) works, and how storage devices work, and how all of them work together at very low levels.
- The Build Process: You will need knowledge of compilers (GCC), linkers, and libraries (Glibc) to compile the source code into a running system.
- Kernel Configuration: There are thousands of configuration options for the Linux kernel. It is no small task to determine which ones should be enabled for your specific hardware.
- Debugging is extremely difficult. If you make a small mistake, the system will most probably fail to boot. Very often, the error will not be clear. Most debugging will involve days of your life trying to figure out the source of the problem.

c) Is anyone in India working in this field? Find at-least three to four engineers.

The Indian engineering community is growing rapidly and making significant contributions to the Linux kernel and other open-source operating systems. The majority of engineers work out of the Indian research and development offices for companies like IBM, Oracle, Red Hat, and Samsung, among several large tech company offices. Here are a few such contributors:

- Balbir Singh: Mr. Singh is currently employed by Meta (formerly Facebook) and is one of the more prominent kernel developers from India. He gained notoriety for his pioneering work on control groups (cgroups), which is a kernel feature that has provided the basis of today's container ecosystem (e.g. Docker).

- Suparna Bhattacharya: Ms. Bhattacharya is a Distinguished Engineer and Linux Kernel SRE Lead at IBM. Suparna is a long-time contributor and has extensive knowledge in file systems and the storage stack of the kernel.

- Aneesh Kumar K.V.: Mr. Kumar has been with IBM Linux Technology Center in Bengaluru, India, and has contributed considerably in many areas of the kernel including memory management, and the file system.

- K. Prasad: Mr. Prasad formed works at Texas Instruments and has made significant contributions to power management and CPU scheduling for the kernel.