Department of Cybersecurity

Linux Programming(24CY23XX)

ASSIGNMENT -1

Name: Mohd Sahil Ansari Class:3A

USN:ENG24CY0134 Roll no:22

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

Answer:

Linux is an open-source operating system that is based on the Unix operating system architecture. It was originally developed by Finnish software engineer Linus Torvalds in 1991. Linux combines the Linux kernel with various software packages and utilities to create complete operating system distributions. The system is freely available and allows users to view, modify, and distribute its source code under open-source licenses like the GNU General Public License (GPL).

Three Advantages of Linux:

1. **Open Source and Cost-Effective:** Linux is freely available to download, use, and modify without requiring expensive license fees. This makes it highly cost-effective for businesses and individuals. The open-source nature allows developers to customize the system according to their specific needs and contribute to its improvement.
2. **Superior Security and Stability:** Linux is renowned for its robust security features and stability. It has a sophisticated user permissions system that requires administrator authorization for critical operations. The system experiences fewer crashes and can run for extended periods without requiring reboots, making it ideal for servers and critical applications.
3. **High Performance and Resource Efficiency:** Linux can efficiently run on both modern and older hardware systems. It has excellent resource management capabilities and can handle multiple processes simultaneously. The system is lightweight and can be configured to use minimal system resources while delivering optimal performance.

Three Disadvantages of Linux:

1. **Steep Learning Curve:** Linux requires users to have a deeper technical understanding compared to user-friendly systems like Windows or macOS. New users often find the command-line interface challenging, and mastering Linux administration requires significant time and effort.
2. **Limited Software Compatibility:** While Linux has a robust ecosystem of open-source applications, it may lack compatibility with certain proprietary software titles commonly used in business environments. Some specialized commercial software and popular consumer applications are not available for Linux platforms.
3. **Hardware Driver Limitations:** Linux may have limited support for some peripheral hardware devices compared to Windows. Finding and installing appropriate drivers for newer or specialized hardware components can sometimes be challenging, especially for consumer-oriented devices.

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

Answer:

Feature	Linux	Mac	Android	Windows
Source Code	Open-source, freely available for modification	Closed-source, proprietary	Open-source (AOSP) but proprietary additions	Closed-source, proprietary
Cost Structure	Free to use and distribute	Premium pricing, tied to Apple hardware	Free for manufacturers, revenue from services	Requires paid licenses for most versions
Hardware Compatibility	Runs on diverse hardware platforms and architectures	Exclusive to Apple hardware ecosystem	Primarily mobile devices, tablets, and embedded systems	Extensive compatibility with various PC hardware
User Interface	Highly customizable desktop environments (GNOME, KDE, etc.)	Unified, polished interface with limited customization	Touch-optimized interface with home screens and widgets	Traditional desktop interface with Start menu and taskbar

Feature	Linux	Mac	Android	Windows
Security Model	Strong permissions system, community-driven security updates	Tight hardware-software integration, secure ecosystem	App sandboxing, Google Play security, permission-based	Built-in antivirus, Windows Defender, frequent security updates
Application Ecosystem	Extensive open-source software repository	Curated App Store with quality control	Google Play Store with millions of apps	Massive software library including gaming and enterprise applications
Target Audience	Developers, servers, technical users, enterprise environments	Creative professionals, premium consumers, education	Mobile users, general consumers	Business users, gamers, general consumers, enterprise
File System	Hierarchical structure starting from root (/), ext4, Btrfs	HFS+, APFS, hierarchical Unix-like structure	Linux-based file system with Android-specific partitions	Drive-based system (C:, D:), NTFS, FAT32
Command Line Interface	Powerful terminal with Bash shell by default	Terminal with Bash/Zsh shell	Limited command line access (requires root)	Command Prompt and PowerShell
Gaming Support	Growing support, Steam Deck, compatibility layers	Limited gaming library, focus on casual games	Extensive mobile gaming ecosystem	Dominant gaming platform with DirectX support

---

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

Answer:

Three Technical Reasons for Linux Preference in Mainframe Servers for Legacy Applications:

1. **Superior Stability and Reliability (RAS - Reliability, Availability, Serviceability):** Linux provides exceptional system stability that can handle continuous operation for months or years without requiring reboots. The kernel's robust error handling and recovery mechanisms ensure that system failures in one component don't cascade to

affect the entire system. This is critical for legacy applications that were designed to run continuously and cannot tolerate unexpected downtime. Linux's proven track record of handling high workloads and maintaining system uptime makes it ideal for mission-critical legacy systems.

2. **Massive Concurrent Transaction Processing Capabilities:** Linux excels at managing large-scale, simultaneous transactions which is essential for legacy mainframe applications. The system can process up to one trillion web transactions daily with built-in capacity scaling and shared memory architecture for direct application communication. Legacy applications often require handling numerous concurrent database operations, file system access, and network communications. Linux's advanced process scheduling, memory management, and I/O handling capabilities ensure that these legacy workloads continue to perform optimally even under heavy load conditions.
3. **Backward Compatibility and Legacy Code Support:** Linux provides excellent support for running older applications and legacy code without requiring extensive modifications. The system maintains compatibility with various programming languages, libraries, and runtime environments that legacy applications depend on. Unlike proprietary systems that may deprecate older APIs or require costly migrations, Linux's open-source nature ensures that legacy applications can continue running with necessary compatibility layers. Additionally, Linux supports multiple processor architectures and can run on both modern and older hardware platforms, making it easier to maintain legacy systems without complete hardware overhauls.

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

Answer:

The Linux File System follows the Filesystem Hierarchy Standard (FHS), which defines a tree-like directory structure starting from the root directory (/). This hierarchical organization ensures consistency across different Linux distributions and provides a standardized way to organize system and user files.

Linux File System Structure:

text

```
/
├── bin/          # Essential command binaries (ls, cat, mkdir)
├── boot/         # Boot loader files and kernel images
├── dev/          # Device files for hardware components
├── etc/          # System-wide configuration files
```

```
├── home/         # User home directories
|   ├── user1/
|   └── user2/
├── lib/          # Shared libraries for system programs
├── lib32/        # 32-bit shared libraries
├── lib64/        # 64-bit shared libraries
├── media/        # Mount points for removable media (USB, CD/DVD)
├── mnt/          # Temporary mount points for filesystems
├── opt/          # Optional software packages
├── proc/         # Virtual filesystem with system information
├── root/         # Root user's home directory
├── run/          # Runtime data for system processes
├── sbin/         # System administration binaries
├── srv/          # Data directories for system services
├── sys/          # Virtual filesystem for system hardware info
├── tmp/          # Temporary files (cleared on reboot)
├── usr/          # User utilities and applications
|   ├── bin/      # Non-essential command binaries
|   ├── lib/      # Libraries for /usr/bin programs
|   ├── local/    # Local software installations
|   ├── sbin/     # Non-essential system binaries
|   └── share/    # Architecture-independent data
└── var/          # Variable data files
    ├── log/      # System log files
    ├── cache/    # Application cache data
    └── tmp/      # Temporary files that survive reboots
```

Key Directory Explanations:

- **/ (Root):** The top-level directory from which all other directories branch out

- **/bin:** Contains essential command binaries needed for system maintenance and recovery
- **/etc:** Stores system-wide configuration files and shell scripts
- **/home:** Contains personal directories for regular users
- **/usr:** Secondary hierarchy containing the majority of user utilities and applications
- **/var:** Contains files that vary in size during system operation, such as logs and databases
- **/proc:** Virtual filesystem providing information about running processes and kernel parameters
- **/dev:** Contains device files that represent hardware components as file interfaces

---

Question 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)

Answer:

Red Hat's Business Model and Revenue Generation:

Red Hat has successfully demonstrated that companies can generate substantial revenue from open-source software through innovative business models. In 2012, Red Hat became the first open-source technology company to surpass $1 billion in annual revenue, and in 2019, IBM acquired Red Hat for approximately $34 billion, validating the commercial viability of open-source business models.

Primary Revenue Streams:

1. **Subscription-Based Services (87% of Revenue):** Red Hat's primary revenue model is subscription-based, where customers pay for enterprise-grade support, regular updates, security patches, and professional services. According to Red Hat's FY2023 financial reports, approximately 87% of their revenue comes from subscription services. Customers gain access to stable, tested versions of open-source software with guaranteed support and maintenance.
2. **Enterprise Support and Professional Services:** Red Hat provides 24x7 technical support, consulting services, training programs, and certification courses. These services help organizations maximize their open-source investments and ensure smooth deployment and operation of complex enterprise systems. Professional services, while representing a smaller portion (around 13% of revenue), provide high-value consulting and implementation support.
3. **Training and Certification Programs:** Red Hat offers comprehensive training courses and industry-recognized certifications for system administrators, developers, and IT

professionals. These educational services create an additional revenue stream while building expertise in Red Hat technologies within the market.

Market Analysis and Success Factors:

Value Addition Strategy: Red Hat doesn't just distribute free software; they add significant value through rigorous testing, integration, and quality assurance. They take multiple open-source projects and integrate them into cohesive, enterprise-ready platforms. This integration work, combined with extensive testing and validation, justifies the subscription costs.

Enterprise Trust and Compliance: Large organizations require vendor support, service level agreements, compliance certifications, and legal indemnification. Red Hat provides these enterprise necessities that free, community-supported versions cannot offer. This creates a sustainable market for commercial open-source solutions.

Red Hat's Unique Position: Red Hat succeeded because they were early pioneers who established strong community relationships and built comprehensive product portfolios spanning operating systems, virtualization, cloud computing, and container technologies. Their success is attributed to their principled approach of keeping all software completely open-source while monetizing through services and support.

Market Reality: Most modern open-source companies have adopted different models than Red Hat's pure open-source approach. Companies like MongoDB, Elastic, and HashiCorp use open-core models with proprietary enterprise features, as this approach provides better revenue predictability and investor appeal.

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

Answer:

The command to display the current system date and time in Linux is:

bash

date

Sample Output:

text

Fri Sep 21 20:21:00 IST 2025

Alternative Commands and Options:

1. **Using timedatectl (for systemd systems):**

bash

timedatectl

2. **Custom date format:**

bash

date +"%Y-%m-%d %H:%M:%S"

Output: 2025-09-21 20:21:00

3. **Display date in specific timezone:**

bash

TZ='UTC' date

4. **Display date with day of the year:**

bash

date +"%Y-%m-%d %H:%M:%S (Day %j of the year)"

The basic date command without any options displays the current system date and time in the default format, showing the day of the week, month, date, time, timezone, and year.

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

Answer:

The command used to check how long the system has been running is:

bash

uptime

Sample Output:

text

21:54:11 up 13 days, 4:29, 1 user, load average: 0.21, 0.21, 0.12

Output Explanation:

- **21:54:11:** Current time when the command was executed
- **up 13 days, 4:29:** System has been running for 13 days, 4 hours, and 29 minutes
- **1 user:** Number of users currently logged into the system
- **load average:** System load averages for the past 1, 5, and 15 minutes respectively

Additional uptime Command Options:

1. **Pretty format:**

bash

uptime -p

Output: up 13 days, 4 hours, 29 minutes

2. **Show boot time:**

bash

uptime -s

Output: 2025-09-08 17:25:00

3. **Alternative commands:**

bash

w

This command also shows uptime information along with logged-in users.

bash

cat /proc/uptime

This shows uptime in seconds (total uptime and idle time).

The uptime command is essential for system administrators to monitor system availability, assess when the last reboot occurred, and evaluate system stability.

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

Answer:

Both shutdown -h now and halt are used to stop the system, but they differ in their approach and execution methods:

shutdown -h now:

- **Graceful Shutdown Process:** Initiates a controlled shutdown sequence that properly terminates all running processes and services
- **User Notification:** Broadcasts warning messages to all logged-in users about the pending shutdown
- **Process Management:** Allows running processes to save their data and terminate gracefully
- **Service Termination:** Executes shutdown scripts to properly stop network services and system daemons
- **Login Prevention:** Blocks new user login attempts during the shutdown process
- **File System Safety:** Ensures all file systems are properly unmounted and buffers are flushed to disk

- **System State:** Changes the system run level in an orderly manner before powering off

halt:

- **Immediate System Stop:** Directly stops the Linux kernel and all CPU functions without waiting for processes to terminate
- **No Graceful Termination:** Does not execute shutdown scripts or allow processes to save data
- **Aggressive Approach:** Forcefully halts all system operations immediately
- **No User Warning:** Does not notify logged-in users about the impending halt
- **System State:** On modern systems, may power off completely, but traditionally leaves the system powered on in a halted state
- **Risk Factor:** Higher risk of data loss or file system corruption due to abrupt termination

Key Differences:

1. **Safety:** shutdown -h now is safer as it ensures data integrity, while halt may cause data loss
2. **Process Handling:** shutdown -h now allows graceful process termination, halt forcibly kills all processes
3. **User Experience:** shutdown -h now provides user notifications and grace periods, halt does not
4. **System Services:** shutdown -h now properly stops services, halt terminates them abruptly
5. **Data Protection:** shutdown -h now protects against data corruption, halt may cause file system issues

Recommendation: Always use shutdown -h now for normal system shutdown operations to ensure data safety and system integrity. Use halt only in emergency situations or when immediate system termination is required.

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

Answer:

Both init 0 and shutdown -h are used to halt the system, but they operate through different mechanisms and offer varying levels of safety.

init 0:

- **Run Level Change:** Changes the system run level to 0 (halt/shutdown mode)

- **System Process:** Executes through the init system (SysV init or systemd)
- **Service Termination:** Stops services according to the configured run level scripts
- **Process Handling:** Terminates processes in an orderly fashion based on init scripts
- **Immediate Execution:** Initiates shutdown immediately without user notification or grace periods
- **Legacy Compatibility:** Traditional Unix-style command that works across different Unix-like systems

shutdown -h:

- **Scheduled Shutdown:** Allows scheduling shutdown for a specific time (default is 1 minute)
- **User Notification:** Broadcasts warning messages to all logged-in users
- **Grace Period:** Provides time for users to save work and log out gracefully
- **Flexible Options:** Supports various options for timing and user messages
- **Login Prevention:** Creates /run/nologin file to prevent new logins
- **Modern Implementation:** Designed specifically for multi-user environments

Safety Comparison:

shutdown -h is SAFER for the following reasons:

1. **User Notification System:** shutdown -h warns all logged-in users about the pending shutdown, allowing them to save their work and log out properly. init 0 provides no such warning.
2. **Grace Period:** shutdown -h typically provides a default grace period (usually 1 minute) before actual shutdown begins, giving users time to react. init 0 executes immediately.
3. **Login Prevention:** shutdown -h prevents new users from logging in during the shutdown process, maintaining system integrity. init 0 doesn't implement this protection.
4. **Cancellation Option:** shutdown -h can be canceled using shutdown -c if needed, while init 0 cannot be stopped once initiated.
5. **Multi-User Environment Safety:** In environments with multiple users, shutdown -h is much safer as it considers the needs of all active sessions.

Technical Analysis:

- **Data Protection:** Both commands allow processes to terminate gracefully, but shutdown -h provides better data protection through its notification and grace period system.
- **System Services:** Both properly terminate system services, but shutdown -h does so in a more controlled manner.
- **Emergency Situations:** init 0 might be preferred in emergency situations where immediate shutdown is required, but shutdown -h now can achieve the same immediate effect while still providing basic safety measures.

Conclusion:
shutdown -h is significantly safer than init 0 for normal shutdown operations because it prioritizes user notification, data safety, and system integrity. It should be the preferred method in multi-user environments and production systems where data protection and user experience are important considerations.

Question 10: A system administrator accidentally powers off a Server machine without shutting it down properly. What problems can occur to the said Server? (CO2)

Answer:

When a system administrator accidentally powers off a server without proper shutdown, several serious problems can occur that may affect system integrity, data consistency, and service availability:

File System and Data Corruption Issues:

1. **File System Corruption:** Abrupt power loss can corrupt the file system structure, leading to inconsistent metadata, damaged inodes, and broken directory structures. This may render parts of the file system unreadable or require extensive repair procedures using tools like fsck.
2. **Data Loss and Inconsistency:** Applications that were writing data at the time of power loss may lose unsaved information. Database transactions in progress may be left in an inconsistent state, leading to data corruption or referential integrity problems.
3. **Journal and Log Corruption:** Modern file systems use journaling to maintain consistency, but sudden power loss can corrupt the journal itself, making recovery more complex and potentially leading to data loss.

System Boot and Recovery Problems:

4. **Boot Failure:** The system may fail to boot properly due to corrupted system files, damaged boot loader configuration, or inconsistent file system state. This could require manual intervention and recovery procedures.

5. **Kernel Module Issues:** Kernel modules that were being loaded or unloaded during power loss may become corrupted, leading to kernel panics or module loading failures during the next boot.
6. **Configuration File Corruption:** Critical system configuration files that were being updated may become corrupted or contain incomplete information, affecting system services and functionality.

Service and Application Impact:

7. **Database Corruption:** Database systems like MySQL, PostgreSQL, or Oracle may suffer from corrupted tables, incomplete transactions, or damaged log files. Recovery may require restoring from backups or performing complex database repair procedures.
8. **Application State Loss:** Running applications lose their current state, cached data, and temporary files. Services may fail to restart properly or may start with inconsistent configurations.
9. **Network Service Disruption:** Network services may not restart correctly, leading to connectivity issues, broken SSL certificates, or misconfigured network interfaces.

Hardware-Related Consequences:

10. **Disk Drive Damage:** Sudden power loss can potentially damage hard drives, especially older mechanical drives, due to improper head parking or incomplete write operations.
11. **Memory Corruption:** System memory contents are lost, and any pending write operations from memory to disk are not completed, potentially causing data inconsistency.

Recovery and Mitigation Requirements:

12. **Extended Downtime:** Recovery from improper shutdown often requires significantly more time than a normal reboot, including file system checks, service verification, and data integrity validation.
13. **Manual Intervention:** System administrators may need to manually start services, verify configurations, and perform data recovery procedures, extending the outage duration.
14. **Backup Restoration:** In severe cases, administrators may need to restore data from backups, potentially losing recent changes and requiring coordination with users and dependent systems.

Preventive Measures:

- Implement Uninterruptible Power Supply (UPS) systems
- Use proper shutdown procedures and scripts

- Enable file system journaling and regular backups
- Configure database systems with appropriate transaction logging
- Implement monitoring systems to detect and prevent improper shutdowns
- Train staff on proper shutdown procedures and emergency protocols

Long-term Impact:
Repeated improper shutdowns can lead to cumulative damage, reduced system reliability, and increased maintenance requirements. The server may become more prone to failures and may require more frequent maintenance and monitoring to ensure continued operation.

Brainstorming Questions:

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

Answer:

Yes, it is absolutely possible to build your own operating system using the Linux kernel as a foundation. The open-source nature of Linux provides several pathways for creating custom operating systems:

Linux Distribution Development: The most practical approach is creating a Linux distribution, which combines the Linux kernel with selected software packages, utilities, and configurations. Hundreds of Linux distributions exist, from Ubuntu and Fedora to specialized distributions like Kali Linux for penetration testing or embedded distributions for IoT devices.

Custom Kernel Compilation: You can download the Linux kernel source code, modify it according to your requirements, and compile a customized version. This allows you to add new features, remove unnecessary components, optimize for specific hardware, or implement custom security measures.

From-Scratch Operating System: While more challenging, you can use Linux as inspiration to build a completely new operating system. Projects like MenuetOS or TempleOS demonstrate that individuals can create functional operating systems, though this requires extensive knowledge of computer architecture, hardware interfaces, and system programming.

Legal Framework: The GPL (General Public License) under which Linux is released allows you to use, modify, and distribute the code freely, as long as you make your modifications available under the same license. This legal framework enables innovation and customization without licensing restrictions.

---

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

Answer:

Building an operating system, even based on the Linux kernel, involves numerous technical, resource, and practical challenges:

Technical Challenges:

1. **Deep Technical Expertise Required:** Developing an OS requires comprehensive knowledge of computer architecture, hardware interfaces, memory management, process scheduling, device drivers, and low-level system programming. The learning curve is extremely steep.
2. **Hardware Compatibility:** Ensuring compatibility across different hardware platforms, processors, and peripheral devices requires extensive testing and driver development. Modern hardware complexity makes this particularly challenging.
3. **Device Driver Development:** Creating drivers for graphics cards, network interfaces, storage devices, and other hardware components is complex and time-consuming. Each hardware vendor has different specifications and requirements.
4. **System Integration:** Integrating thousands of software components, libraries, and utilities into a cohesive, stable system requires sophisticated build systems and dependency management.

Resource and Scale Challenges:

5. **Massive Codebase:** The Linux kernel alone contains millions of lines of code. Managing, understanding, and modifying such a large codebase requires significant resources and expertise.
6. **Testing and Quality Assurance:** Ensuring system stability, security, and performance across different hardware configurations requires extensive testing infrastructure and time investment.
7. **Continuous Maintenance:** Operating systems require ongoing maintenance, security updates, bug fixes, and feature development. This demands sustained effort and resources.

Community and Ecosystem Challenges:

8. **Software Ecosystem:** Creating a viable operating system requires supporting applications, development tools, and user software. Building or porting this ecosystem is enormously challenging.
9. **User Adoption:** Convincing users to adopt a new operating system over established alternatives requires compelling advantages and extensive marketing efforts.
10. **Hardware Vendor Support:** Getting hardware manufacturers to provide drivers and support for a new operating system is extremely difficult without significant market presence.

Financial and Business Challenges:

11. **Funding Requirements:** Developing and maintaining an operating system requires substantial financial investment for development teams, infrastructure, and ongoing operations.
12. **Market Competition:** Competing against established operating systems like Windows, macOS, and existing Linux distributions requires significant competitive advantages.

---

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

Answer:

Yes, several Indian engineers and organizations are actively working in the field of operating system development and Linux kernel contributions:

1. Kumar Priyansh - BackSlash Linux Developer
A 20-year-old developer from Madhya Pradesh who single-handedly created BackSlash Linux, one of the newest Linux distributions developed in India. His distribution combines Ubuntu and Debian platforms with KDE and GNOME integration. Priyansh started creating OpenSUSE-based distributions in 2011 and released BackSlash Linux in November 2016, demonstrating that individual developers can create professional-grade Linux platforms.

2. Dr. V. Kamakoti - IIT Madras Director and BharOS Lead
Currently serving as the Director of Indian Institute of Technology (IIT) Madras, Dr. Kamakoti led the development of BharOS (Bharat Operating System), an indigenous mobile operating system based on early Linux versions. BharOS was successfully tested by Union ministers in 2023 and represents a significant effort to create a "Made in India" operating system with enhanced security protocols and customization for Indian requirements.

3. Aakarsh Jain - Samsung Semiconductor India R&D
A kernel developer working at Samsung Semiconductor India R&D, specializing in Linux kernel livepatch technology. He has contributed to deep kernel-level development, particularly in areas of kernel runtime patching and security updates. His work focuses on enabling Linux machines to receive security fixes without rebooting, which is crucial for critical infrastructure and high-availability systems.

4. Subhajeet Mukherjee - Operating System Researcher
A 21-year-old programmer and author studying Computer Science at Foothill College, with a patent pending in both the U.S. and India regarding a new operating system design. He has worked on creating a hybrid kernel operating system that combines both monolithic and microkernel architectures. His research addresses user experience issues and security monitoring structures, and he received recognition from former President Barack Obama for his innovative work.

Government and Institutional Efforts:

BOSS GNU/Linux Development Team: The Centre for Development of Advanced Computing (C-DAC) has developed BOSS (Bharat Operating System Solutions), an Indian Linux distribution based on Debian. The latest stable release is BOSS 10.0 (Pragya), released in March 2024. This represents a coordinated government effort to promote indigenous operating system development.

Indian Linux Kernel Contributors: Several Indian engineers contribute to the mainline Linux kernel through companies like Red Hat India, Intel India, and various startups. These contributors work on diverse areas including device drivers, security frameworks, memory management, and performance optimization.

These examples demonstrate that India has active participation in operating system development, ranging from individual innovators to government-sponsored initiatives and corporate research teams. The field continues to grow with increasing emphasis on indigenous technology development and open-source contributions.
