

**Experiment-1**

Operating Systems

**Program:**

MCA

**SVKM'S-NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**School of Technology Management and Engineering**

**[2024-25]**

LAB Manual

PART A

(PART A: TO BE REFFERED BY STUDENTS)

**Experiment No.01**

**A.1 Aim:**

Conduct the Study of Unix Operating System. Also perform the following tasks.

1. **Identify the Difference between the Unix with other operating system**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **UNIX** | **WINDOWS** |
| Kernel Type | Monolithic kernel | Hybrid kernel |
| File System | Supports various file systems like UFS, ext, and ZFS. | Primarily uses NTFS, with support for FAT32 and exFAT. |
| User Interface | command-line interface (CLI), with modern Unix-like systems offering GUIs. | Graphical User Interface (GUI) oriented, with CLI options (PowerShell, Command Prompt). |
| Security | Strong user permissions and multi-user capabilities designed from the ground up. | Historically had more security vulnerabilities but has improved significantly. |
| Licensing | Many variants are open-source (e.g., BSD, Linux), but some are proprietary (e.g., AIX, Solaris). | Proprietary software developed by Microsoft |

|  |  |  |
| --- | --- | --- |
| Parameter | UNIX | LINUX |
| Origin | Originated at AT&T Bell Labs in the early 1970s. | Created by Linus Torvalds in 1991 as a Unix-like operating system. |
| Source Code | Original Unix source code is proprietary, but many Unix-like systems are open-source (e.g., BSD). | Completely open-source and available under the GPL license. |
| Development | Developed and maintained by commercial vendors | Developed collaboratively by a global community of developers. |

1. **Identify the Pros and Cons of using Unix or similar like OS**

Pros:

* Stability and Reliability: Unix systems are reliability.
* Security: Strong architecture with advanced file permissions and encryption capacity.
* Performance: Powerful resource management suitable for high-performance computing tasks.
* Portability: Can run on a wide range of platforms.
* Flexibility: Powerful scripting and programming capabilities for customization.

Cons:

* Complexity: Steep learning curve, especially for users unfamiliar with command-line interfaces.
* Software Availability: Some popular software may not be readily available on Unix.
* Hardware Support: lack support for some newer or niche hardware out of the box.
* Cost: Certain Unix variants (e.g., AIX, Solaris) can be expensive compared Linux.

1. **Identify The features of Unix OS**

Multi-User Capability: Supports multiple users accessing the system.

Multitasking: Allows execution of multiple processes.

Portability: Adaptable to different hardware architectures.

Security: Advanced file permissions, encryption, and user management.

Networking: Built-in support for networking with robust capabilities.

Hierarchical File System: Organized directory structure.

Shell: Powerful command-line interface (CLI) with scripting capabilities.

Utilities: Standard tools and applications for various tasks like text editing, compiling, and networking.

Libraries: Precompiled routines for common tasks, enhancing application development.

1. **Identify the Components of Unix OS**

Kernel: Core part that manage hardware, system calls, memory, and process management.

Shell: Command-line interpreter for user interaction with the kernel.

File System: Manages files, directories, and access control mechanisms.

Utilities: Standard tools and applications for system operations.

Libraries: Precompiled routines used by applications to perform common tasks.

1. **Identify the Architecture of Unix or similar OS**

Hardware: Physical components of the computing system.

Kernel: Core of the OS managing resources and providing system services.

System Call Interface: Allows user applications to request services from the kernel.

Shell and Utilities: User-level interface and tools for system management and operations.

User Applications: Software and programs that users interact with and run on top of the OS.

**A.2 Theory:**

**Introduction to Unix Operating System**

**Historical Context and Development**

Unix was developed in the 1960s and 1970s at AT&T's Bell Labs by Ken Thompson, Dennis Ritchie, and others. It was initially conceived as a flexible, multi-tasking system for programmers, aimed at providing a comfortable computing environment. Unix's simplicity, portability, and powerful features made it a significant advancement over its predecessors.

* **1969**: Ken Thompson wrote the first version of Unix in assembly language for a PDP-7 minicomputer.
* **1970s**: Unix was rewritten in the C programming language by Dennis Ritchie and Thompson, enhancing its portability and allowing it to run on various hardware platforms. This rewrite was a pivotal moment, as it made Unix one of the first operating systems that could be easily adapted to different machines.

**Philosophy and Design Principles**

Unix was designed with several key principles in mind:

* **Simplicity**: Keep the system simple and understandable. Unix tools and commands are designed to perform one task well.
* **Modularity**: Build complex tasks by combining simpler tools. The use of pipelines to connect commands exemplifies this modular approach.
* **Portability**: Make the system easy to move to different hardware platforms. Writing Unix in C was crucial for this goal.
* **Multi-user capability**: Support multiple users simultaneously, sharing resources without interference.
* **Multitasking**: Allow multiple processes to run concurrently, enhancing productivity and resource utilization.

**Impact and Influence**

Unix has profoundly influenced the development of other operating systems and computing in general:

* **Standardization**: Unix introduced many concepts and standards that became foundational in computing, such as the hierarchical file system, the concept of processes, and the use of plain text for configuration.
* **Birth of Open Source**: The licensing of Unix led to the development of numerous Unix-like systems. In 1983, the GNU Project was launched to create a free Unix-like operating system. Later, in 1991, Linus Torvalds released the Linux kernel, leading to the creation of the Linux operating system, which combined GNU tools with the Linux kernel.
* **Educational Tool**: Unix has been widely used in academic settings for teaching operating system concepts and programming.
* **Server and Enterprise Use**: Due to its stability, reliability, and performance, Unix and Unix-like systems have been dominant in server environments and enterprise computing.

**Evolution and Variants**

Over the decades, Unix has evolved into many different versions and variants, each with unique features and optimizations:

* **BSD (Berkeley Software Distribution)**: Developed at the University of California, Berkeley, BSD introduced many innovations, including the TCP/IP networking stack.
* **System V**: Developed by AT&T, System V introduced features like the System V init process, which has influenced many Unix-like systems.
* **Commercial Unixes**: Various companies have developed their own versions of Unix, such as IBM's AIX, HP's HP-UX, and Sun Microsystems' Solaris.
* **Linux**: Although not a direct Unix derivative, Linux is a Unix-like system that embodies many of the same principles and designs, making it a popular choice for servers, desktops, and embedded systems.

**Unix Today**

Today, Unix and its variants remain crucial in many areas of computing:

* **Servers**: Unix-like systems power a significant portion of the internet's infrastructure, providing the backbone for web servers, databases, and other critical services.
* **Supercomputing**: Unix-like systems are used in many of the world's fastest supercomputers.
* **Embedded Systems**: Unix-like operating systems are found in a wide range of embedded systems, from networking equipment to consumer electronics.
* **Desktop and Development Environments**: Unix and Unix-like systems provide robust environments for software development, scientific computing, and general desktop use.

The principles and design philosophies of Unix continue to influence modern operating systems and computing practices, ensuring its legacy will endure for many years to come.

**Difference between Unix and Other Operating Systems**

**Unix vs Windows**

* **Kernel Type**:
  + Unix: Monolithic kernel, which means the entire operating system works in a single large block of code that runs as a single process with a single address space.
  + Windows: Hybrid kernel, combining aspects of microkernel and monolithic design.
* **File System**:
  + Unix: Uses various file systems such as UFS, ext, and ZFS.
  + Windows: Primarily uses NTFS, with support for FAT32 and exFAT.
* **User Interface**:
  + Unix: Predominantly command-line interface (CLI), though modern Unix-like systems (like Linux) offer GUIs.
  + Windows: Graphical User Interface (GUI) oriented, with a less prominent CLI (PowerShell, Command Prompt).
* **Security**:
  + Unix: Designed with security as a core component, featuring strong user permissions and multi-user capabilities.
  + Windows: Historically had more security vulnerabilities, though recent versions have significantly improved.
* **Licensing**:
  + Unix: Many variants are open-source (e.g., BSD, Linux), but some are proprietary (e.g., AIX, Solaris).
  + Windows: Proprietary software developed by Microsoft.

**Unix vs Linux**

* **Origin**:
  + Unix: Originated at AT&T Bell Labs in the early 1970s.
  + Linux: Created by Linus Torvalds in 1991 as a Unix-like operating system.
* **Source Code**:
  + Unix: Original Unix source code is proprietary, though many Unix-like systems are open source (e.g., BSD).
  + Linux: Completely open source and available under the GPL license.
* **Development**:
  + Unix: Developed and maintained by commercial vendors (e.g., IBM's AIX, Oracle's Solaris).
  + Linux: Developed collaboratively by a global community of developers.

**Pros and Cons of Using Unix or Similar OS**

**Pros:**

* **Stability and Reliability**: Unix systems are known for their long uptimes and reliability.
* **Security**: Unix’s architecture makes it a secure system, with strong user permissions and encryption capabilities.
* **Performance**: Efficient resource management makes Unix suitable for high-performance computing tasks.
* **Portability**: Unix can run on a wide range of hardware platforms.
* **Flexibility**: Powerful scripting and programming capabilities provide significant customization options.

**Cons:**

* **Complexity**: Steep learning curve for new users, especially those unfamiliar with command-line interfaces.
* **Software Availability**: Some popular software applications may not be available on Unix.
* **Hardware Support**: Unix might not support some newer or niche hardware out of the box.
* **Cost**: Certain Unix variants (e.g., AIX, Solaris) can be expensive compared to free Unix-like systems like Linux.

**Features of Unix OS**

* **Multi-User Capability**: Multiple users can access the system simultaneously.
* **Multitasking**: Supports concurrent running of multiple processes.
* **Portability**: Can be adapted to run on different types of hardware.
* **Security**: Advanced file permissions and encryption features.
* **Networking**: Robust networking capabilities, essential for servers.
* **Hierarchical File System**: Organized directory structure.
* **Shell**: Powerful command-line interface with scripting capabilities.

**Components of Unix OS**

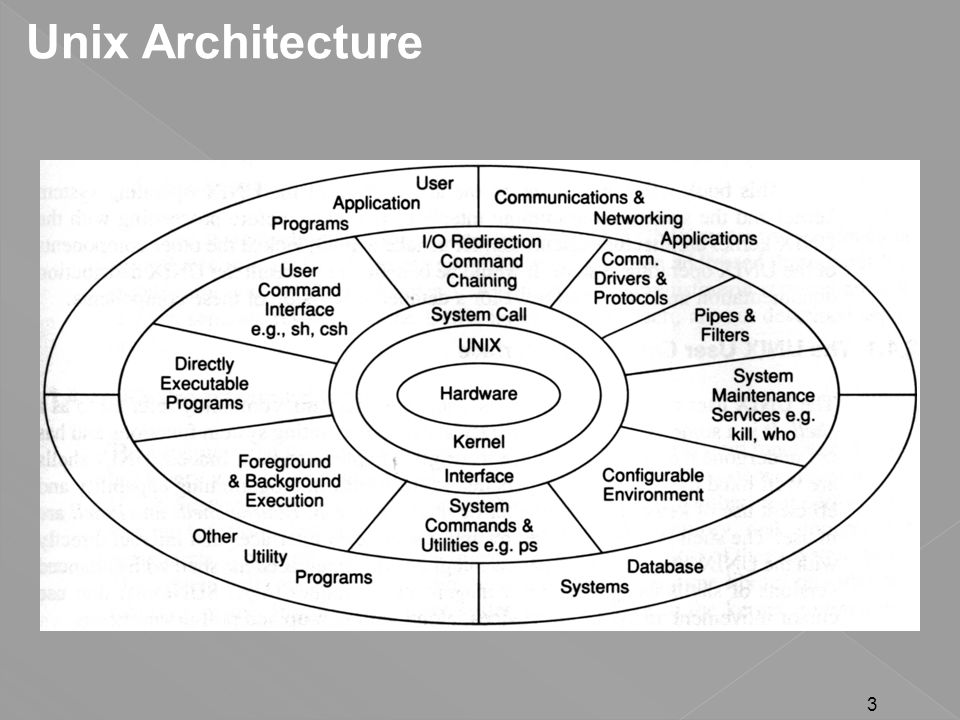
* **Kernel**: The core part of Unix, managing hardware, system calls, memory, and process management.
* **Shell**: The command-line interpreter that allows users to interact with the kernel.
* **File System**: Manages files and directories, with access control mechanisms.
* **Utilities**: Standard tools and applications for performing various tasks, such as text editors, compilers, and network tools.
* **Libraries**: Precompiled routines that programs can use to perform common tasks without re-writing code.

**Architecture of Unix or Similar OS**

The architecture of Unix and Unix-like operating systems is typically layered as follows:

1. **Hardware**: The physical machine components.
2. **Kernel**: The core part of the OS, managing system resources, memory, processes, and hardware devices.
3. **System Call Interface**: Provides the means for user applications to interact with the kernel.
4. **Shell and Utilities**: The command-line interface and standard utilities that provide user-level operations and management.
5. **User Applications**: Software and applications that users run on top of the OS.

**Diagram of Unix Architecture**



This architecture ensures a clear separation between user operations and system-level operations, providing stability, security, and flexibility.

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments as per the submission instructions.)***

| Roll No. A073 | Name: Aryan Srivastava |
| --- | --- |
| Class: FY MCA | Batch: B3 |
| Experiment Number-1 | |
| Date of Experiment: | Date of Submission: |
| Grade: |  |

**B.1 Program with Output to be written by student**

**B.2 Answers the following question based on study**

1. **Who were the primary developers of the Unix operating system, and where was it developed?**

Unix was developed by Ken Thompson and Dennis Ritchie at AT&T's Bell Labs in the 1960-70s.

They were supported by a group of researchers and programmers who contributed to its initial versions and subsequent developments.

1. **What are the core design principles of Unix, and how do they contribute to its flexibility and power?**

Simplicity: Tools and commands perform one task well.

Modularity: Complex tasks are built by combining simpler tools, exemplified by the use of pipelines.

Portability: Rewriting Unix in C made it adaptable to different hardware platforms.

Multi-user capability: Supports multiple users sharing resources simultaneously.

Multitasking: Allows multiple processes to run concurrently, enhancing productivity

1. **Why was the rewriting of Unix in the C programming language a significant milestone in its development?**

Rewriting Unix in the C programming language made it accessable across different platforms.

C allowed Unix to be easily compiled and adapted, this made portability a practical goal.

1. **How does Unix support multi-user capability, and why is this feature important?**

Unix supports multi-user capability through its design of user accounts and permissions. Each user has a unique id and a set of permissions that control access to files and resources.

This feature is important for environments where multiple users need real time sharing of data without interference.

1. **Explain the concept of multitasking in Unix. How does Unix handle multiple processes simultaneously?**

Unix allows multitasking by managing multiple processes simultaneously. Each process runs independently and can execute its tasks at a time with other processes. The operating system schedules and allocates resources, ensuring optimal performance.

1. **Describe the hierarchical file system in Unix. How does it organize and manage files and directories?**

The hierarchical file system in Unix organizes files and directories in a tree-like structure.

Every file and directory has a unique path starting from the root directory which allows for easy navigation, storage management, and access control using file permissions.

1. **What is the role of the shell in Unix, and how does it enhance user interaction with the system?**

The shell in Unix is a command-line interpreter that acts as a user interface to the kernel.

It allows users type commands. The shell also supports scripting, which can be used to automate tasks and create workflows.

1. **How has Unix contributed to the development of networking protocols and standards, particularly TCP/IP?**

Through BSD variants, Unix played a significant role in the development of networking protocols like TCP/IP.

These protocols became fundamentals for internet communication and networking standards.

1. **What are some of the major Unix variants, and how do they differ from each other in terms of features and use cases?**

Unix has evolved into various versions, including BSD (Berkeley Software Distribution), System V, commercial Unixes like AIX, HP-UX, and Solaris, and Unix-like systems such as Linux.

Each variant differs in features, licensing, and targeted use cases, catering to diverse computing needs from servers to embedded systems.

1. **In what areas of computing is Unix still widely used today, and what are the reasons for its continued relevance?**

Servers: Powering critical internet infrastructure, web servers, and databases.

Supercomputing: Running on some of the world's fastest computers.

Embeddedsystems: Found in networking equipment, consumer electronics, and industrial devices.

Desktop and development environments: Providing platforms for software development, scientific computing, and general use.