**SIMATS SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CHENNAI-602105**

**The UNIX time- sharing system**

**A CAPSTONE PROJECT REPORT**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION TECHNOLOGY**

**Submitted by**

**Tharun Kumar T.R (192211541)**

**Gnanendra C (192211158)**

**Under the Supervision of**

**Dr E.K.Subramanian**

**MARCH 2024**

**DECLARATION**

We, **Tharun Kumar T R, Gnanendra C**, students of **‘Bachelor of Engineering in Information Technology**, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the work presented in this Capstone Project Work entitled **The UNIX time- sharing system** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics.

(Tharun Kumar T R 192211541)

(Gnanendra C 192211158)

Date:

Place:

**CERTIFICATE**

This is to certify that the project entitled **“The UNIX time- sharing system**

**”** submitted by **Tharun Kumar T R, Gnanendra C** has been carried out under our supervision. The project has been submitted as per the requirements in the current semester of B. Tech Information Technology.

Teacher-in-charge

**Dr E.K.Subramanian**

**Table of Contents**

|  |  |
| --- | --- |
| **S.NO** | **TOPICS** |
| 1 | **Abstract** |
| 2 | **Introduction** |
| 3 | **Problem Statement** |
| 4 | **Proposed Design**   1. Requirement Gathering and Analysis 2. Tool selection criteria 3. Scanning and Testing Methodologies |
| 5. | **Functionality**   1. **Scalability and Resource Management** 2. Containerization and Virtualization 3. Developer Tools and DevOps Integration |
| 6 | **UI Design**   1. Layout Design 2. Feasible Elements Used 3. Elements Positioning and Functionality |
| 7 | **Conclusion** |

**ABSTRACT:**

The UNIX time-sharing system is a pillar of the operating system community, representing core ideas that have influenced contemporary computer environments. This abstract explores the core concepts of UNIX, including its history design ideas effects. UNIX was developed as a simple yet effective operating system in the Bell Labs in the late 1960s with the goal of promoting a cooperative computing environment and effective resource usage. Fundamentally, UNIX is modular, straightforward, and adaptable—it personifies the Unix tenet of "doing one thing well." The time-sharing design of the UNIX paradigm is fundamental to its operation, as it allows several users to access the system at once and interact with individual computers.

**Introduction:**

An important turning point in the history of computers was the introduction of the UNIX time-sharing system, which signalled a revolution in operating system functionality and design. Created in the rich environment of Bell Labs during the late 1960s by developers Dennis Ritchie, Ken Thompson, and associates, UNIX became a trailblazing example of efficiency, simplicity, and modularity. We set out on a quest to discover the fundamentals of UNIX in this introduction, following its history, clarifying its tenets, and scrutinizing its lasting influence in modern computer environments.

When computers first came into being, most of the systems were batch-oriented and monolithic, and they were connected to big mainframe computers. There were few options for customization or interaction, and users had to submit their jobs to centralized processing units via heavy terminals. With its vision of a computing environment characterized by user empowerment, collaborative engagement, and effective resource usage, UNIX evolved as a radical break from this paradigm.

About 40 installations of PDP-11 UNIX have been placed into operation since it started operating in February 1971; these installations are typically smaller than the system that is being discussed here. The majority of them work on tasks like preparing and formatting textual materials and patent applications, gathering and processing trouble data from different Bell System switching equipment, and entering and verifying phone service orders. Our own installation is mostly used for document writing and study on computer science subjects like as operating systems, languages, and networks.

**Problem Statement:**

Legacy UNIX systems encounter numerous difficulties in the current computer environment, despite their long history and crucial role in establishing modern computing paradigms. The preservation and adaption of legacy UNIX installations create significant challenges for enterprises looking to modernize their IT infrastructure in order to satisfy changing needs for security, agility, and scalability. This issue statement outlines the main obstacles to updating legacy UNIX systems and suggests strategies for successfully overcoming them.

**Proposed Design:**

**Requirements Gathering and Analysis:**

1. Identify key stakeholders involved in the modernization effort, including IT management, system administrators, application owners, and end-users.
2. Conduct a comprehensive assessment of the existing legacy UNIX systems, including hardware specifications, software configurations, network architecture, and security posture.

**Tool Selection Criteria:**

1. Compatibility with legacy UNIX systems: Ensure that selected tools are compatible with existing UNIX environments, including support for legacy hardware architectures and operating system versions.
2. Security features: Prioritize tools that offer robust security features, including encryption, access controls, intrusion detection, and compliance auditing capabilities.

**Scanning and Testing Methodology:**

1. Conduct an inventory of legacy UNIX systems, including hardware specifications, software configurations, network topology, and dependencies.
2. Utilize discovery tools, network scanners, and configuration management databases (CMDBs) to identify and catalog all assets within the UNIX environment**.**

**Functionality:**

**Scalability and Resource Management: -**

Dynamic resource allocation: Enable dynamic scaling of computing resources, including CPU, memory, and storage, to accommodate fluctuating workloads and optimize resource utilization.

**Containerization and Virtualization: -**

Container support: Enable portability, isolation, and scalability by integrating containerization technologies, such Docker or Kubernetes, to encapsulate legacy UNIX applications and dependencies.

**Developer Tools and DevOps Integration: -**

Development environments: To make application development, debugging, and testing on UNIX platforms more efficient, provide developer-friendly environments and tools including IDEs, version control systems, and testing frameworks.

**User Experience and Accessibility: -**

User interfaces: To make UNIX systems easier to use, more productive, and more accessible, give system administrators, developers, and end users access to intuitive user interfaces and command-line interfaces (CLIs).

**Monitoring and Performance Management: -**

Performance monitoring: Provide comprehensive monitoring and performance management capabilities to track system performance metrics, resource utilization, and application health status in real-time.

**Architectural Design:**

**Microservices Architecture:**

* Decompose legacy UNIX applications into modular, loosely-coupled microservices to enable independent development, deployment, and scaling.
* Utilize lightweight communication protocols, such as RESTful APIs or message queues, to facilitate inter-service communication and interoperability.

**Security and Compliance Controls:**

* Implement defense-in-depth security controls, including network segmentation, encryption, and least privilege access, to protect data and mitigate security risks in legacy UNIX environments.
* Integrate security monitoring tools, such as intrusion detection/prevention systems (IDS/IPS), vulnerability scanners, and SIEM platforms, to detect and respond to security incidents in real-time.

**Observability and Monitoring:**

* Instrument applications with distributed tracing, logging, and metrics collection mechanisms to enable comprehensive observability and monitoring of system health, performance, and reliability.
* Utilize observability tools, such as Prometheus, Grafana, or ELK Stack, to aggregate, visualize, and analyze telemetry data from legacy UNIX systems and microservices architectures.

**UI Design:**

**Dashboard:**

Overview of system status including CPU usage, memory usage, disk space, and number of active users.

**File Management**: -

File browser interface resembling a directory tree structure.

* Ability to navigate through directories, view file properties, and perform basic file operations (e.g., create, delete, copy, move).

**User Management**:

* Interface for managing user accounts, groups, and permissions.
* Ability to add, delete, or modify user accounts and their associated attributes.

**Feasible Element Used:**

**Multi-User Environment:**

The UNIX time-sharing system allows multiple users to access the system concurrently, with each user having their own session and workspace.

**Process Management:**

The system manages multiple processes concurrently, allowing users to run multiple programs simultaneously. It provides utilities for process creation, termination, suspension, and monitoring.

Networking:

* UNIX systems have built-in networking capabilities, allowing users to connect to remote systems and access network resources.
* This includes utilities for network configuration, communication, and remote access (e.g., SSH)..

**Element Positioning and Functionality:**

**User Menu and Account Management:**

* **Positioning**:
* Located within the menu bar or navigation area.
* **Functionality**:
* Allows users to access account settings, manage user profiles, and perform user-related tasks (e.g., change password, update profile information, log out). Provides options for user authentication and switching between user accounts.
  + .

**Process Manager:**

* **Positioning**:
* Positioned alongside or below the file browser, within the same window or pane.
* **Functionality**:
* Displays a list of active processes, including their process IDs, resource usage, and status. Allows users to manage processes (e.g., terminate, suspend) and view detailed process information.

**Help and Documentation:**

* **Positioning:**
* Accessible through a dedicated help menu or button within the UI.
* **Functionality**:
* Provides access to system documentation, user manuals, and help resources. Offers assistance and guidance on using the system, troubleshooting common issues, and accessing support channels.

**Conclusion:**

The UNIX time-sharing system is proof of the lasting influence of UNIX concepts and their fundamental significance in forming contemporary computing environments. Since its inception, the UNIX time-sharing system has enabled users to cooperate, develop, and efficiently handle a wide range of computing activities thanks to its multi-user capabilities, sturdy shell interface, and modular design.   
  
In spite of changing technological environments, the UNIX time-sharing system has managed to stay relevant and flexible thanks to its sophisticated architecture and commitment to fundamental UNIX principles like simplicity, modularity, and interoperability. Its hierarchical file system and process management features allow for easy navigation and multitasking, and its shell interface offers users a strong and adaptable environment for system interaction.

**REFERENCES:-**

**.** Ritchie, Dennis M., and Ken Thompson. "The UNIX time‐sharing system." Bell System Technical Journal 57.6 (1978): 1905-1929.

. Ritchie, D. M., & Thompson, K. (1978). The UNIX time‐sharing system. Bell System Technical Journal, 57(6), 1905-1929.

. Ritchie, Dennis M., and Ken Thompson. "The UNIX time-sharing system." Communications of the ACM 17.7 (1974): 365-375.

. Ritchie, D. M., & Thompson, K. (1973). The UNIX time-sharing system. ACM SIGOPS Operating Systems Review, 7(4), 27.

. Bourne, Stephen Richard. "UNIX time-sharing system: The UNIX shell." The Bell System Technical Journal 57.6 (1978): 1971-1990.