VirtuOS: Mini Operating System Simulation Report

Muhammed Emir Daloğlu / 210402041
Ali Kerem Erhan / 210402021
Emre Kılıç / 210402027

Github URL: https://github.com/emirdaloglu/VirtuOS

1. Introduction

VirtuOS is an educational operating system simulation designed to demonstrate the core principles of modern operating systems in an interactive, user-friendly environment. The project features a modular architecture, a modern graphical user interface (GUI), and a comprehensive set of features covering process management, memory management, concurrency, and file system operations. VirtuOS is intended as both a learning tool and a demonstration of best practices in OS design.

---

2. Design Overview

2.1. Architecture

VirtuOS is implemented in Python and organized into four main modules:

- **Process Management** (process.py)
- **Memory Management** (memory.py)
- **Concurrency & Synchronization** (concurrency.py)
- **File System** (filesystem.py)

A central GUI (gui.py) built with Tkinter provides a modern, tabbed interface for interacting with all OS components. Each module is self-contained, with clear interfaces and responsibilities, making the system easy to extend and maintain.

2.2. User Interface

The GUI is the primary interface for VirtuOS. It features:

- Tabbed navigation for each OS component
- Consistent, modern dark theme with accent colors and icons
- Equally sized, well-aligned buttons for all actions
- Tooltips and a status bar for user feedback
- Output areas for displaying results and system state

This design ensures that users can intuitively explore OS concepts without being overwhelmed by command-line complexity.

---

3. Meeting OS Principles

3.1. Process Management

VirtuOS implements a full process management subsystem:

- **Process Control Block (PCB):** Each process is represented by a PCB storing its PID, name, state, and power profile.
- **Schedulers:** Supports FIFO, Round Robin, Multi-Level Feedback Queue (MLFQ), and Power-aware scheduling. Users can switch schedulers at runtime.
- **Multi-core Simulation:** Users can set the number of CPU cores, and the system simulates parallel process execution.
- **Process State Transitions:** Users can create, switch, and terminate processes, with all state changes visualized in the GUI.

OS Principles Demonstrated:

- Process abstraction and isolation
- Scheduling algorithms and fairness
- Context switching and state management
- Multi-core and parallelism concepts

3.2. Memory Management

VirtuOS simulates realistic memory management:

- **Paging:** Each process can have a page table, and address translation is supported.
- **Memory Visualization:** Users can see frame allocation and fragmentation.
- **Swapping:** Processes can be swapped in and out, simulating virtual memory.
- **Fragmentation Analysis:** The system visualizes memory fragmentation, helping users understand internal and external fragmentation.

OS Principles Demonstrated:

- Virtual memory and paging
- Address translation
- Memory allocation and fragmentation
- Swapping and memory constraints

3.3. Concurrency & Synchronization

VirtuOS includes a concurrency module:

- **Producer-Consumer Problem:** Simulated with a buffer, producer, and consumer actions.
- **Locks and Condition Variables:** Implemented to manage access to shared resources.

- **Thread Simulation:** While not using real OS threads, the system models thread-like behavior and synchronization.

OS Principles Demonstrated:

- Concurrency and race conditions
- Synchronization primitives (locks, conditions)
- Classical problems (producer-consumer)

3.4. File System

VirtuOS features a hierarchical file system:

- **Directories and Files:** Users can create, read, write, and delete files and directories.
- **Permissions:** Each file has user/admin permissions (r/w/x).
- **Encryption:** Files can be marked as encrypted.
- **Search and Visualization:** Recursive file search and directory tree visualization are provided.

OS Principles Demonstrated:

- File and directory abstraction
- Access control and permissions
- File operations and navigation
- File system structure and search

---

4. Theme and Design Decisions

4.1. Theme: Modern, Educational, and User-Friendly

The theme of VirtuOS is to make OS concepts accessible and engaging:

- **Modern GUI:** The dark theme, icons, and consistent layout make the system inviting and easy to use.
- **Visualization:** Every major OS concept is visualized—process queues, memory frames, file system trees—helping users "see" what's happening.
- **Interactivity:** All actions are performed via buttons and dialogs, lowering the barrier for experimentation.

4.2. Design Choices Influenced by Theme

- **Tabbed Interface:** Each OS component is separated into a tab, mirroring how real OSes modularize their subsystems.
- **Status Bar and Tooltips:** Immediate feedback and explanations help users understand the effects of their actions.
- **Consistent Button Layout:** All actions are equally accessible, reinforcing the idea that OS components are peers in the system.

- **Icons and Branding:** The name “VirtuOS” and the use of icons give the project a professional, memorable identity

---

5. Extensibility and Future Work

VirtuOS is designed to be extensible:

- New scheduling algorithms, memory models, or file system features can be added with minimal changes.
- The GUI can be further enhanced with drag-and-drop, real-time graphs, or even a web-based interface.
- More advanced concurrency (e.g., real Python threads) or networking could be added for deeper exploration.

6. Conclusion

VirtuOS successfully demonstrates the core principles of operating systems in a modern, interactive, and educational way. By combining a modular backend with a polished GUI, it makes complex OS concepts accessible to students and educators alike. The project’s theme of clarity, interactivity, and professionalism is reflected in every design decision, from the code structure to the user interface.VirtuOS is not just a simulation—it’s a platform for learning, experimentation, and inspiration in the world of operating systems.

Appendix: How to Run See the README for instructions. Launch with `python3 gui.py` for the full VirtuOS experience.