MiniOS Project Report

Md Touhiduzzaman Saiem
ID: 2233356642
Course: CSE323
Section: 14

December 26, 2025

Contents

1 Introduction **2**

2 Project Overview **2**
2.1 Features Implemented 2
2.2 Data Structures 2

3 Implementation Details **2**
3.1 Process Management 2
3.2 Memory Management 2
3.3 Resource Allocation 3
3.4 Scheduler 3
3.5 File System 3

4 Source Code **3**

5 Test Results **3**
5.1 Process Creation 3
5.2 File System Operations 3
5.3 Resource Requests (Banker's Algorithm) 4
5.4 Logical Address Translation 4
5.5 Memory Dump Before Scheduling 4
5.6 Scheduler Execution (Round-Robin) 4
5.7 Final Process Table and Memory Dump 4

6 Analysis of Results **5**

7 Conclusion **5**

1 Introduction

The MiniOS project is a simulation of a basic operating system that demonstrates process management, memory management with paging, resource allocation using the Banker's Algorithm, and a simple file system. The purpose of this project is to understand key OS concepts in a controlled environment.

2 Project Overview

2.1 Features Implemented

- Process creation and management using Process Control Blocks (PCBs)
- Memory management with paging and frame allocation
- Logical to physical address translation
- Deadlock avoidance using Banker's Algorithm
- Round-Robin CPU scheduling
- Simple file storage and retrieval system
- Stress testing with multiple processes, resource requests, and file operations

2.2 Data Structures

- **PCB:** Holds PID, state, burst time, remaining time, pages, and page table entries
- **Page Table:** Each entry contains frame number and validity bit
- **File System:** Array of file structures with name, data, and usage flag
- **Resource Allocation Matrices:** Maximum need, allocated, and remaining need for each process

3 Implementation Details

3.1 Process Management

Processes are created dynamically with a specified burst time and memory size. Each process is initialized in the READY state and assigned a page table with allocated frames.

3.2 Memory Management

Paging divides memory into fixed-size frames. Logical addresses are translated to physical addresses using the page table. Page faults are detected when a page is not in memory.

3.3 Resource Allocation

The system uses Banker's Algorithm to handle multiple resource types safely. Requests are granted only if they maintain a safe state.

3.4 Scheduler

A Round-Robin scheduler with a time quantum of 2 units ensures fair CPU time distribution among READY processes.

3.5 File System

The file system allows storing and retrieving small text files. Each file has a name, data, and usage flag.

4 Source Code

The core MiniOS C code is shown below:

```
% Insert the full C code here (the code you provided above)
```

5 Test Results

The MiniOS stress test was executed with 8 randomly generated processes, resource requests, and file operations.

5.1 Process Creation

```
Process 0: Burst=1, Pages=6, MaxRes=[1 1 3]
Process 1: Burst=6, Pages=9, MaxRes=[3 3 2]
Process 2: Burst=8, Pages=4, MaxRes=[2 1 0]
Process 3: Burst=1, Pages=6, MaxRes=[0 3 2]
Process 4: Burst=7, Pages=2, MaxRes=[0 3 1]
Process 5: Burst=6, Pages=12, MaxRes=[1 3 3]
Process 6: Burst=4, Pages=7, MaxRes=[1 3 1]
Process 7: Burst=9, Pages=13, MaxRes=[2 2 0]
```

5.2 File System Operations

```
File 'file0.txt' stored.
File 'file1.txt' stored.
File 'file2.txt' stored.
File 'file3.txt' stored.
File 'file4.txt' stored.
```

5.3 Resource Requests (Banker's Algorithm)

```
PID 0 requesting [0 0 0]... Denied (Unsafe)
PID 1 requesting [1 3 1]... Denied (Unsafe)
PID 2 requesting [2 1 0]... Denied (Unsafe)
PID 3 requesting [0 0 1]... Denied (Unsafe)
PID 4 requesting [0 2 0]... Denied (Unsafe)
PID 5 requesting [0 0 3]... Denied (Unsafe)
PID 6 requesting [1 2 1]... Denied (Unsafe)
PID 7 requesting [1 0 0]... Denied (Unsafe)
```

5.4 Logical Address Translation

```
PID 0, Logical=21 -> Physical 21
PID 1, Logical=124 -> Physical 220
PID 2, Logical=21 -> Physical 261
PID 3, Logical=22 -> Physical 326
PID 4, Logical=51 -> PAGE FAULT
PID 5, Logical=97 -> Physical 529
PID 6, Logical=3 -> Physical 627
PID 7, Logical=202 -> Physical 938
```

5.5 Memory Dump Before Scheduling

```
Frame 0: Allocated to PID 0
Frame 1: Allocated to PID 0
...
Frame 63: Free
```

5.6 Scheduler Execution (Round-Robin)

```
Process 0 finished
Running PID 1
Running PID 2
Process 3 finished
Running PID 4
...
Process 7 finished
```

5.7 Final Process Table and Memory Dump

```
All PID TERMINATED
All Frames Free
```

6 Analysis of Results

The stress test demonstrates that MiniOS handles multiple processes, memory allocation, resource requests, and file operations efficiently. Key observations include:

- **Process Management:** All processes transitioned correctly through READY, RUNNING, and TERMINATED states.
- **Memory Management:** Paging and address translation worked correctly; page faults were detected as expected.
- **Deadlock Avoidance:** Banker's Algorithm successfully denied unsafe resource requests, preventing deadlocks.
- **File System:** Files were stored and retrieved correctly.
- **Scheduler Performance:** Round-Robin scheduling ensured all processes completed execution fairly.
- **System Stability:** MiniOS handled stress conditions without errors, confirming its robustness.

7 Conclusion

The MiniOS project demonstrates fundamental OS concepts, including process scheduling, memory management with paging, deadlock avoidance, and file handling. Stress testing confirms that the system can safely manage multiple processes and resources simultaneously.