Final Project: MiniOS: Building a Simplified Operating System Simulator

DESCRIPTION

You will build a MiniOS simulator in Java that mimics key functionalities of a modern operating system. Rather than operating at the hardware level, your program will simulate core OS components such as process scheduling, memory allocation, file systems, and synchronization using high-level constructs.

This is a modular, multi-phase project designed to help you apply OS concepts to practical code.

GOALS

1. Understand the architecture of an operating system.
2. Implement core OS components in Java.
3. Simulate process control blocks, memory segments, and semaphores.
4. Build a command-line shell that interacts with the MiniOS modules.

TASKS

The project is divided into 3 modules. Arrange your time to ensure on-time completion.

Week-by-Week Breakdown:

Week	Module	Topics
0.5	Process Management	PCB, process creation, and Round Robin scheduling
1	Memory Management	Memory array simulation and First-Fit allocation
1.5	Synchronization & CLI Shell	Semaphores, shared resources, shell commands
2	Testing	Combine modules, fix bugs, test workflows
3	Presenting	Present and demonstrate your project

The project structure:

MiniOS/

├── *Main.java* *// CLI Shell*

├── *PCB.java* *// Process Control Block*

├── *ProcessManager.java* *// PCB and Scheduler*

├── *MemoryManager.java* *// Memory Allocation*

├── *Semaphore.java* *// Synchronization*

Module 1: Process Management

Concepts: PCB (Process Control Block), scheduling algorithms

1. Create a `PCB` class:
   - Fields: `pid`, `name`, `state`, `active`
   - States: `READY`, `RUNNING`, `BLOCKED`
2. Create a `ProcessManager` class:
   - Method `createProcess(String name)`
   - Method `schedule()` that prints each READY process as RUNNING (simulate Round Robin or other CPU scheduling algorithms)
   - Make a process to sleep for a while to simulate its running time, such as 5 seconds
3. Add support to `listProcesses()` to display all processes and their states.

Module 2: Memory Management

Concepts: memory allocation and deallocation

1. Create a `MemoryManager` class:
   - Use a fixed-size array to simulate memory blocks (e.g., `int[] memory = new int[100]`)
2. Add method `allocate(int pid, int size)`:
   - Use First-Fit, Best-Fit, or Worst-fit to find a free region and store the PID in that region.
3. Add method `free(int pid)` to release memory held by a process.
4. Add method `printMemory()` to display current memory layout.

Module 3: Synchronization & CLI Shell

Concepts: Semaphores, mutual exclusion, command parsing

1. Implement a `Semaphore` class with `waitSem()` and `signal()` methods using `synchronized`.

   Himt: `waitSem()` is to implement `acquire()` and `signal()` is to implement `release()`

2. Create `Main.java` to act as the CLI shell:
   - Support commands:
     - `create [name]`
     - `ps`
     - `schedule`
     - `alloc [pid] [size]`
     - `mem`
     - `exit`
3. Use `Scanner` to read input and `switch` or `if-else` to process commands.

Optional Bonus 1: Deadlock Handling

- Simulate deadlock scenarios.
- Implement a simple deadlock detection or avoidance strategy.

Optional Bonus 2: Implement virtual paging and memory visualization.

Optional Bonus 3: GUI interface

Run your miniOS: (You don't need to use the same commands or print out the exact same printings)

```
C:\Users\wuya\OneDrive - University of Wisconsin - Platteville\Desktop\UW\CS3230 OS\Spring2024\assignments\project-java\sample solution>javac *.java

C:\Users\wuya\OneDrive - University of Wisconsin - Platteville\Desktop\UW\CS3230 OS\Spring2024\assignments\project-java\sample solution>java Main
Welcome to MiniOS!
MiniOS> create game
MiniOS> ps
PID 0 [game] - READY
MiniOS> alloc 0 10
Allocated at 0
MiniOS> mem
1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MiniOS> create study
MiniOS> ps
PID 0 [game] - READY
PID 1 [study] - READY
MiniOS> alloc 1 20
Allocated at 10
MiniOS> mem
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MiniOS> schedule
Running: PID 0 [game] - RUNNING
Running: PID 1 [study] - RUNNING
MiniOS> exit

C:\Users\wuya\OneDrive - University of Wisconsin - Platteville\Desktop\UW\CS3230 OS\Spring2024\assignments\project-java\sample solution>
```

SUBMISSION

1. **Source Code** with clear structure and comments.
2. **Presentation** to explain and demonstrate the project.
3. **User Manual** or CLI usage guide.

Rubric

Criteria	Ratings		Pts
Completion: all required documents are submitted	**10 pts** All are submitted.	**0 pts** A required submission, such as screenshots, source code, or report, is missing	10 pts

Criteria	Ratings				Pts
Module 1: Process Management	**20 pts** Correct	**0 pts** Not correct or no submission			20 pts
Module 2: Memory Management	**20 pts** Correct	**0 pts** Program is not working or no submission			20 pts
Module 3: CLI	**20 pts** Correctly create a new process	**0 pts** Not correct or no submission			20 pts
Bonus 1: Deadlock	**10 pts** Correct	**0 pts** Not correct or no submission			10 pts
Bonus 2: Paging	**10 pts** Correct	**0 pts** Not correct or no submission			10 pts
Bonus 3: GUI	**10 pts** Correct	**0 pts** Not correct or no submission			10 pts
Presentation: Clearly explain the project and successfully demonstrate it	**30 pts** Clearly explain the project and successfully demonstrate it	**20 pts** Explain the project fine or demonstration has minor errors	**10 pts** major errors	**0 pts** Not correct or no submission	30 pts
Total Points: 100+30					

CONGRATULATIONS, YOU'VE COMPLETED FINAL PROJECT!