Operating Systems Concepts

Assignment 1 – Design Programming Project SharkOS

CSIS 3810

Samuel Torres

10/23/2023

**Introduction**

The SharkOS project is a powerful operating system simulator that mirrors real-world OS intricacies. It seamlessly integrates single-tasking and multitasking capabilities through a robust Process Control Block (PCB) system. This OS employs a round-robin algorithm, ensuring efficient multitasking while handling interrupts seamlessly. Programs' instructions are read and placed in the Request Queue, where processes are executed within allocated CPU time or managed effectively. The system intelligently saves process states, enabling seamless resumption from the last execution point. SharkOS provides a concise yet immersive glimpse into operating system functionality.

**Project Overview:**

The SharkOS project encompasses six essential files, serving as the backbone for simulating a multitasking operating system. Here's an overview of the key components:

**1. SharkMachine Class:**

* Acts as the interface between the user and the OS, simulating the machine's startup.
* Users can initiate the project by running java SharkMachine in the command line.

**2. PCB Class:**

* Represents the Process Control Block, storing crucial process information.
* Properties include id, registers (holding instructions and addresses), CPU time, process state, and other vital data.

**3. RequestQueue Class:**

* Manages the PCB objects, enqueuing processes in the order of their arrival and dequeuing processes if necessary.
* Handles the logic for efficient process management, ensuring processes execute within the allocated time.

**4. Scheduler Class:**

* Determines arrival times and quantum time (QT or TQ) for processes.
* Utilizes a round-robin algorithm, making decisions based on processes' execution times and arrivals.
* Simulates multitasking by swiftly switching between process contexts and making necessary decisions.

**5. FileHandler Class:**

* Handles file reading and writing operations.
* Reads program instructions and inserts them into the OS' memory.
* Writes the content of registers to corresponding output files.

**6. SharkOS Class:**

* Instantiates and integrates all components (PCB, Scheduler, RequestQueue, FileHandler, and memory) to form the OS.
* Implements the Round Robin algorithm, a vital multitasking component.
* Processes programs, determining their arrival times and executing them within the specified CPU time.
* Manages interruptions, saving process states and efficiently resuming tasks.

**Functionality Highlights:**

* Identification of program files within the "programs" folder, creating corresponding PCB objects.
* Assignment of random arrival times to processes, mapped using the Scheduler class.
* Dynamic loading of processes into the request queue based on their arrival times.
* Execution of processes with evaluation against the time quantum and management of interruptions.
* Identification of opcodes and addresses, invoking specific functions (e.g., ADD, SUB, LDI) to update process properties.
* Detailed output generation, capturing registers and memory states at each instruction, stored in the "outputs" folder.

**Project Directory Structure:**

The SharkOS project follows a structured directory organization for easy navigation and management:

**Parent Folder:**

**src**: Contains all Java source files (.java), compiled class files (.class), program and output folders.

* SharkMachine.java
* PCB.java
* RequestQueue.java
* Scheduler.java
* FileHandler.java
* SharkOS.java

**programs**: Stores program files (.txt) to be processed by the OS.

* program1.txt
* program2.txt
* ...

**outputs**: Holds output files generated during program execution.

* program1-output.txt
* program2-output.txt
* ...

**Example:** A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Simulation Overview:**

In this simulation, the Shark Machine OS is run through a series of processes specified in the provided terminal commands. Each process is defined by a text file (e.g., program1.txt, program2.txt) and arrives at the system at a certain time, with a designated burst time indicating how long it requires to execute. The Shark Machine's Round Robin scheduling algorithm handles the execution of these processes. The quantum time allocated to each process in the Request Queue is set at 3 seconds.

Execution Details:

1. **Process 5 (program5.txt):**
   * Arrived at 1 second with a burst time of 5 seconds.
   * Executed and completed, taking 4 seconds.
   * Process state saved and removed from the queue.
2. **Process 3 (program3.txt):**
   * Arrived at 6 seconds with a burst time of 10 seconds.
   * Execution started and paused with 7 seconds remaining.
   * Process 5 arrives (from the beginning) at 1 second with a burst time of 2 seconds.
   * Process 5 executes and completes in 2 seconds.
   * Process 5 removed from the queue.
   * Process 3 resumes, taking 1 second to complete.
   * Process 3 removed from the queue.
3. **Process 6 (program6.txt):**
   * Arrived at 7 seconds with a burst time of 1 second.
   * Executed and completed in 1 second.
   * Process 6 removed from the queue.
4. **Process 1 (program1.txt):**
   * Arrived at 8 seconds with a burst time of 1 second.
   * Executed and completed in 1 second.
   * Process 1 removed from the queue.
5. **Process 4 (program4.txt):**
   * Arrived at 9 seconds with a burst time of 9 seconds.
   * Execution started and paused with 6 seconds remaining.
   * Process 5 arrives (from the beginning) at 1 second with a burst time of 2 seconds.
   * Process 5 executes and completes in 2 seconds.
   * Process 5 removed from the queue.
   * Process 4 resumes and completes in the remaining 4 seconds.
   * Process 4 removed from the queue.
6. **Process 2 (program2.txt):**
   * Arrived at 10 seconds with a burst time of 4 seconds.
   * Executed and completed in 1 second.
   * Process 2 removed from the queue.

**Output and Completion:**

* The simulation logs each process's execution and state changes, displaying the queue's updates at every step.
* All processes completed successfully.
* The OS status message indicates the completion of all jobs, and the system queue is empty.

The simulation demonstrates the effective multitasking capabilities of the Shark Machine OS, showcasing its ability to manage and execute multiple processes concurrently within the defined time constraints.