CmpE 322
Operating Systems
Project2-Schedule or not to Schedule?

Soner Kuyar
2021400297

Fall 2023

1 Introduction

Priority scheduling stands as a pivotal method in context of process scheduling, wherein tasks are orchestrated based on their assigned priorities. Unlike other scheduling methods, such as simple round-robin, priority scheduling entails the careful selection of tasks in accordance with their priorities. Every process is assigned a priority, and those with higher priorities are given precedence in execution. In cases of equal priorities, tasks are executed on a round-robin basis.

For this project, the chosen scheduling algorithm for tasks with equal priorities is the round-robin algorithm. This ensures fair and balanced execution among processes with identical priority levels.

In the preemptive variant of priority scheduling, the moment a process arrives in the ready queue, its priority is compared not only with other processes in the ready queue but also with the one currently being executed by the CPU. The process with the highest priority among all available processes is granted CPU access next.

2 Technical Details

2.1 Data Structures

2.1.1 Instruction Struct

The `Instruction` struct encapsulates an instruction's details, including its name and duration.

2.1.2 Process Struct

The `Process` struct encompasses a variety of attributes such as arrival time, burst time, priority, type, and a list of instructions.

2.1.3 ReadyQueue Struct

The `ReadyQueue` struct represents a node in the ready queue, facilitating the sorting of processes based on priority.

2.1.4 ProcessesArrivalQueue Struct

The `ProcessesArrivalQueue` struct is responsible for managing a queue of processes based on their arrival time.

2.2 Functions

2.2.1 createInstruction

The `createInstruction` function generates an instruction with a specified name and duration.

2.2.2 createProcess

The `createProcess` function crafts a process with designated attributes, including its type and priority.

2.2.3 compareTwoProcesses

This function compares two processes, discerning the one with higher priority based on specified rules.

2.2.4 insertReadyQueue

The `insertReadyQueue` function inserts a process into the ready queue according to its priority.

2.2.5 createProcessesArrivalQueue

The `createProcessesArrivalQueue` function initializes an empty arrival queue.

2.2.6 insert_arrival_process

Inserts a process into the arrival queue, organizing them based on their arrival time.

2.2.7 executeProcess

The `executeProcess` function orchestrates the execution of a process, considering its type and updating the current time accordingly.

2.2.8 main

The `main` function orchestrates the entire process, reading instruction durations and process definitions, initializing queues, and executing processes based on the scheduling algorithm.

3 Challenges and Solutions

3.1 Process Upgrades

Handling dynamic process upgrades is a critical aspect of the algorithm. This is achieved by regularly checking the total quantum of a process. Upon reaching a specified threshold, the process is dynamically upgraded to a higher type, ensuring adaptability and resource optimization. Algorithms also handles extreme cases like Platinium upgrade of gold process while has been preempting by other process.

3.2 Equality of Properties

Ensuring fairness in process comparison is essential. The algorithm compares processes based on various properties, including type, priority, arrival time, and name. In cases where priorities are equal algorithm provide a round robin algorithm by sending the current process back to the ready queue with new arrival time.

3.3 Round Robin Algorithm

The algorithm incorporates a round-robin-like behavior by allowing a process to return to the ready queue after exhausting its time quantum. This feature enhances fairness, particularly among processes of the same type, contributing to a more equitable distribution of resources.

The implementation showcases effective management of processes, prioritization based on specific rules, and dynamic handling of process upgrades. The code provides a robust scheduling solution adaptable to diverse scenarios.