**Project 3: Short-Term Process Scheduler**

Kyungchan Im

Grand Canyon University

Course number: CST - 315

Professor Ricardo Citro

Due Apr 2, 2023

**Project 3: Short-Term Process Scheduler**

**[ Introduction ]**

This is the project to demonstrate the scheduling algorithms of multiple process running within one program. The C++ program in question aims to demonstrate the first come first serve scheduling algorithm using three processes: process 1, process 2, and process 3. This algorithm is used in operating systems to manage the execution of multiple processes. The program will analyze the scheduling process by considering the burst time for each process, which represents the amount of time each process requires to complete. Additionally, the program will calculate the waiting time for each process, which is the time each process has to wait before it can start executing. Finally, the program will calculate the average waiting time, which represents the average amount of time that all processes have to wait before starting their execution. This program is an example of how C++ can be used to simulate complex processes and analyze their behavior.

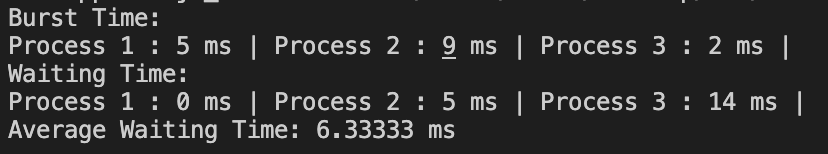
**[ Process Scheduling ]**

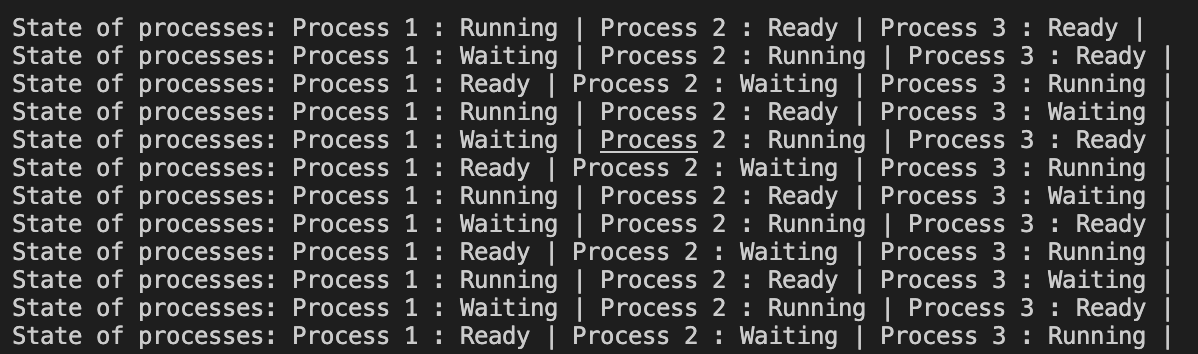
Scheduling is the process of deciding which process or thread should be executed by the CPU at any given time. The CPU can only execute one process at a time, so the scheduling algorithm must decide which process should be executed next, based on a set of predefined rules. The scheduling algorithm takes into account the priority of the process, the amount of CPU time it requires, the time it has been waiting, and other factors, to determine which process should be executed next. The goal of scheduling is to maximize the use of CPU resources, while minimizing the waiting time for processes.

First come first serve (FCFS) scheduling is a simple algorithm used in operating systems to schedule the execution of multiple processes. As the name implies, it prioritizes the process that arrived first, and executes it until completion before moving on to the next process. This approach guarantees that all processes are executed in the order in which they arrived, but it can lead to longer waiting times for processes that arrive later, as they have to wait for the earlier processes to finish.

**[ Analysis ]**

Since FCFS scheduling algorithm executes the process that is allocated to the queue first, the waiting time for each individual process is based on the process in order. Within this program, each process will be assigned with burst time. Three processes are used for this program and each individual burst time is assigned before the program.



Process 1 has 5 ms burst time, Process 2 has 9 ms burst time, and finally Process 3 has 2 ms burst time. Each individual waiting time will be 0 ms for Process 1, 5 ms for Process 2 and lastly 14 ms for Process 3. Once we calculate the average waiting time for individual process, it is 6.33 ms (0 + 5 + 14) / 3 = 19/3 = 6.333 ms.

Once the program executes, the terminal displays which process is in running, ready, and waiting state. First execution with Process 1 running, Process 2 ready, and Process 3 ready. Second Execution express Process 1 waiting since it has already run the cycle and now it is tail of the queue. Process 2 is running since it was the secondly allocated process, and lastly Process 3 is ready state.

**[ Conclusion ]**

In conclusion, the C++ program demonstrated the first come first serve scheduling algorithm by simulating the execution of multiple processes. The program analyzed the scheduling process by considering the burst time and waiting time for each process and calculated the average waiting time. The results showed that the FCFS algorithm prioritizes the process that arrived first, which guarantees the order of execution but may lead to longer waiting times for later processes. The program demonstrated how C++ can be used to simulate complex processes and analyze their behavior, which is essential for optimizing system performance. Overall, the project provided valuable insights into the scheduling algorithms used in operating systems and the importance of efficient process management.