Université de Cergy-Pontoise

# Process Scheduler Simulation

Supported by Mr Philippe Laroque
Made by Giovani Rosario and Aghzou Said
2017

# Introduction

The aim of this project is to simulate the operation of a small process scheduler by using four different algorithms: FiFO, SJF, SRJF and Round Robin. Here is this document are briefs explanations about these algorithms.

# FIFO

FIFO stands for first-in first-out and is also known as first-come-first served (FCFS). Is the simplest scheduling algorithm and is based in put the process in the order that they arrive in a queue, treating always at first the head of the queue and only moving to the next process when the current is finished (Non-preemptive).

The FIFO function in our program contains two inputs :

- ✔ n : number of process
- ✔ bt : burst time array of the processes

And 6 steps :

- ✔ calculate waiting time for each process.
- ✔ calculate turnaround time for each process.
- ✔ display the results.
- ✔ calculate average waiting time.
- ✔ calculate Average Turnaround Time.
- ✔ display the results.

# SJF

Shortest job first is a scheduling algorithm in which the process with the smallest execution time is selected for execution next. Owing to its simple nature, shortest job first is considered optimal. It also reduces the average waiting time for other processes awaiting execution. SFJ is also non-preemptive.

Shortest job first depends on the average running time of the processes. The accurate estimates of these measures help in the implementation of the shortest job first in an environment, which otherwise makes the same nearly impossible to implement. This is because often the execution burst of processes does not happen beforehand. It can be used in interactive environments where past patterns are available to determine the average time between the waiting time and the commands. Although it is disadvantageous to use the shortest-job-first concept in short-term CPU scheduling, it is considered highly advantageous in long-term CPU scheduling. Moreover, the throughput is high in the case of shortest job first.

Shortest job first also has its share of disadvantages. For one, it can cause process starvation for longer jobs if there are a large number of shorter processes. Another is the need to know the execution time for each process beforehand. Often, this is almost impossible in many environments.

In our project, the function **SFJ** contains 2 inputs:

- ➔ n: the number of process
- ➔ bt: burst time array of the processes

And 7 steps:

- ➔ determine the array that contains number of process.
- ➔ sorting burst time in ascending order using selection sort.
- ➔ waiting time for first process will be zero,
- ➔ calculate waiting time for each process.
- ➔ calculate average waiting time.
- ➔ calculate turnaround time for each process..
- ➔ calculate Average Turnaround Time.

# SRJF

This method have the same function of SJF but the difference is that it contains the priority of each process and allows preemption. If a new process having a higher priority than the currently running process arrives, it gets selected immediately. The new process has not to wait until the currently running process finishes.

This function contains 3 inputs :

- ➔ n : number of process
- ➔ bt ; burst time array of the processes
- ➔ pr : priority array of the processes

And 7 steps :

- ➔ determine array of process.
- ➔ sorting burst time, priority and process number in ascending order using selection sort.
- ➔ select waiting time for first process as zero.
- ➔ calculate waiting time for each process.
- ➔ calculate average waiting time.
- ➔ calculate turnaround time for each process..
- ➔ calculate Average Turnaround Time.

# Round Robin

In the Round Robin algorithm the scheduler assigns a fixed time unit per process (quantum-time), and cycles through them. If process completes within that quantum-time it got terminated otherwise it is rescheduled after giving a chance to all other processes.

Our algorithm is based by Five steps:

- ➔ 2. A fixed time is allocated to each process that arrives in the queue. This fixed time is known as the time slot or quantum time.
- ➔ The first process that arrives is selected and sent to the processor for execution. If it fails to complete its execution in the expected quantum of time, then an interruption is generated using an automatic timer.
- ➔ The process is then stopped and is returned to the end of the queue. However, the state is saved and the context is stored in the memory. This helps the process to resume from the point where it was interrupted.
- ➔ The scheduler selects another process from the ready queue and sends it to the processor for execution. It is executed until the Quantum time does not exceed.
- ➔ The same steps are repeated until the entire process is completed.