Final Project - Software Design Document

Andrew Moore

Operating Systems

# Purpose

This program simulates First Come First Serve, Shortest Job First, Priority, and Round Robin scheduling. It then returns the time that the last process finished, throughput in processes per second, the average wait time, and average turnaround time.

# Definitions, Acronyms, and Abbreviations

**Turnaround time** – The amount of time elapsed between when a process was added to the queue and when it was finished. This program calculates the *average* turnaround time by taking the sum of all CPU bursts that have occurred minus each process’s arrival time and divides that value by the total number of processes.

**Throughput** – The number of processes that are completed per second. This program calculates throughput as the total number of processes divided by the total CPU burst time.

**Average Wait Time** – The average amount of time that a process has been waiting to start. The program computes this by summing the total time each process up to process N -1 (where N is the total number of processes) has waited to start since the program started and dividing that number by the total number of processes.

**Average Turnaround Time** – The average amount of time a process has entered the ready queue until it finishes executing. This is calculated by summing the total CPU burst durations and then subtracting the arrival time for each process then dividing by the total number of processes.

**First Come First Serve (FCFS) Scheduling** – Processes are run in the order that they arrive in the ready queue.

**Shortest Job First (SJF) Scheduling** – Processes are run in order of shortest CPU burst time to highest.

**Priority Scheduling** – Processes are run in the order of their priority. 0 is the highest priority.

**Round Robin (RR) Scheduling** – Similar to FCFS but each processes is preempted by a time quantum. If the process’s burst length is longer than the time quantum, the process is put back into the ready queue.

**Time Quantum** – A unit of time used in RR scheduling that defines the maximum amount of time a process can run before the CPU switches to the next process in the ready queue.

**Burst Duration** – The time the CPU spends on a particular process

**Arrival time** – The time the process arrives in the ready queue

**Priority** – Importance of a process. Lower numbers indicate higher priority.

# References

[qsort](http://www.cplusplus.com/reference/cstdlib/qsort/) - Retrieved July 6, 2014

<http://support.microsoft.com/kb/73853>

<http://cs.stackexchange.com/questions/1270/what-is-the-average-turnaround-time> - Retrieved July 9, 2014

Operating System Concepts, 9th Edition - Abraham Silberschatz, Peter B. Galvin, Greg Gagne

# Overall Description

## Software Description and Rationale

This program takes a list of processes and calculates the average wait time, average turnaround time, throughput, and the time the last process finishes using several scheduling algorithms. It is used to demonstrate the differences in performance of various scheduling algorithms.

## Software Features

* Simulates scheduling processes using FCFS, SJF, Priority and RR scheduling.
* Outputs performance statics to the console
* Can take processes as input from a text document

## User Characteristics

This program would be used by someone who wants to explore how different scheduling algorithms preform on a set of processes with known arrival times, priorities, and burst durations. The anticipated use is in an academic setting.

## Constraints

This program looks for its input from a file called input.txt. For this program to execute correctly, the input.txt file must be in the same directory as the executable.

This program must be run from Linux, not a terminal emulator like Cygwin.

## Assumptions and Dependencies

This program looks for its input from a file called input.txt. For this program to execute correctly, the input.txt file must be in the same directory as the executable.

The contents of input.txt must be formatted perfectly for this program to execute. The expected format is as follows:

* The first line must only contain an integer that specifies the number of processes described in the file. There must be n additional lines in the file, where n is the number represented on the first line
* Each subsequent line must contain integer values, delimited by spaces, representing the Process ID; Process Priority; Process Arrival Time; Process CPU Burst Duration in that order.
  + CPU burst duration will be above 200 milliseconds

Figure 1 – Example of input.txt formatting

10

1 7 75 760

2 73 92 200

3 26 107 420

4 82 115 310

5 89 153 340

6 92 174 480

7 17 246 530

8 90 303 280

9 42 328 610

10 78 372 770

# Design Specifics

## Files

Input.txt – a file that stores information about the processes that the program will simulate scheduling.

finalProg\_AMoore.c – the source file for the program.

## Functions

int comparePriority**(**Process **\***elem1**,** Process **\***elem2**)** – Helper function that compares process priorities for sorting.

int compareBurst**(**Process **\***elem1**,** Process **\***elem2**)** – Helper function that compares process burst speeds for sorting.

int compareArrival**(**Process **\***elem1**,** Process **\***elem2**)** – Helper function that compares process arrival time for sorting.

void prioritySch**()** – Runs the simulation for Priority Scheduling and prints the results to the console. Sorts process array by priority.

void firstComeFirstServe**()** – Runs the simulation for FCFS and prints the results to the console. Sorts process array by arrival time

void shortestJobFirst**()** – Runs the simulation for SJF and prints the results to the console. Sorts process array by burst durration

void printProcessList**()** – Prints out a list of all of the processes and their attributes to the console.

float calculateAvgWaitTime**()** – Calculates the average waiting time in milliseconds and returns the answer as a float.

int calculateTimeLastProcFinished**()** – Calculates the time that the last process has finished in milliseconds and returns the answer as an int. Gets the result by adding all of the burst times.

float calculateAvgTurnaroundTime**()** – Calculates the average turnaround in milliseconds and returns the answer as a float.

float calculateThroughput**()** – Calculates the average number of processes completed every second and returns the answer as a float.

# Testing

To test the correctness of the algorithms I created input files that mirrored the examples of each scheduling method in the book and checked the output of my program against the results from the book. Note that each test file only checks the correctness of one particular scheduling method. For example fcfs.txt is only for testing first come first serve.

# Developer’s Guide

This program must be run in a Linux environment and be compiled using the GCC compiler. The correct syntax to compile is as follows:

gcc –o final finalProg\_AMoore.c

This program will not execute correctly in a terminal emulator such as Cygwin. It must be run from Linux. I developed this program in the Ubuntu 14.04 x64 operating system.