La Roche College
SU14-CSCI3040-01 - Operating Systems
Final Project

Due by 6:00 PM on Friday, July 10, 2014

Introduction:

In this final project you are given the opportunity to design and develop a C program that simulates CPU Scheduling Algorithms.

There are 2-parts to this final project, consisting of:

- Software Design Document – worth 30%
- C implementation – worth 70%

Details on requirements for both parts are included in the Requirements-section of this document. Finally, this is an individual project, and the work you submit must be yours and yours only. Any work suspected of plagiarism will be treated according to the La Roche College's Policy on Academic Integrity.

Requirements:

Software Design Document:

Write a high-level Software Design Document (SDD) detailing the design of your C implementation. Your SDD should be clear, concise, and to the point. That said, make sure to cover the following aspects of your C implementation in the SDD (an explanation for a section is provided where deemed necessary):

1. *Purpose* – A high-level non-technical description on what your implementation does.
2. *Definitions, Acronyms, and Abbreviations* – e.g. SJF - Shortest Job First
3. *References* – e.g. Linux Programmer's Manual: exit(3). Retrieved on January 22, 2013.
4. *Overall Description*
    a. *Software Description and Rationale*
    b. *Software Features*
    c. *User Characteristics* – What kinds of users will interact with your implementation
    d. *Constraints* – What constraints does your implementation have; e.g. what kinds of operations it cannot handle
    e. *Assumptions and Dependencies* – What is the set of assumptions and dependencies that your implementation is sure to work on.
5. *Design Specifics*
    a. *Files*
    b. *Functions*
6. *Testing* – Describe what tests have you used to ensure the validity and accuracy of your implementation.
7. *Developer's Guide* – Describe how some other developer can reuse your implementation. What tools (e.g. GNU Compiler Collection) and steps are required to do so?

C Implementation:

Write a C program that simulates CPU Scheduling Algorithms. Your program should support only uniprocessor mode and implement the following scheduling algorithms:

- First Come First Serve (FCFS)
- Non-preemptive Shortest Job First (SJF)
- Non-preemptive Priority (0 to 99 priorities inclusive, where 0 means highest priority)
- Round Robin (RR) with Quantum = 10

As this is a simple CPU scheduling simulator, you're not required to simulate I/O operations, I/O waiting state, interrupts, and/or context switches. However, you should implement your own queue, priority queue, and any other relevant data structure you might need.

Program Input:

Your program will read the information about each process from a file. The first line of the input file should contain a single integer n which represents the number of processes which each of your scheduling algorithms will use. This number will be between 1 and 1000. Following the first line, there will be n lines where each of them will contain 4 integers separated by a single space. The meaning of these integers is as follows:

1. Process ID
2. Process Priority
3. Process Arrival Time (in the Ready Queue)
4. Process CPU Burst Duration -> **Use bursts above 200 milliseconds. It will make the calculations more reasonable**

Do note that the lines in the input file will be ordered by the process's arrival time. That said, an example of the input file would be as follows:

```
10
1 7 75 760
2 73 92 ~~18~~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
```

Program Output:

Your program should run all simulators for the provided input. Once all the simulations are complete, your program should output the following report:

- Name of the scheduler
- Time in the simulation when the last process finished.
- ~~Percent of CPU utilization (the amount of time the CPU is running the processes).~~
- Throughput in process per second.
- Average wait time.

- Average turnaround time.

Program Testing:

To test your program for accuracy use the following C program to generate the relevant program inputs. Do note that to successfully compile this program you need to use the following command: gcc -o gen_scenario gen_scenario.c -std=c99

gen_scenario.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int gen_random( int min, int max ) {
	return min + rand( ) % max;
}

void gen_scenario( int i ) {
	int j = 0;
	printf( "%d\n", i );
	for( int j1 = 0; j1 < i; j1++ ) {
		int k = gen_random( 0, 100 );
		int l = gen_random( 5, 100 );
		int i1 = gen_random( 0, 99 );
		printf( "%d %d %d %d\n", j1 + 1, i1, j + k, l );
		j += k;
	}
}

int main( int argc, char* argv[ ] ) {
	srand( time( NULL ) );
	if( argc > 1 ) {
		gen_scenario( atoi( argv[ 1 ] ) );
	} else {
		gen_scenario( 10 );
	}
	return 0;
}
```

The gen_scenario program will generate process inputs up to the number you've specified in the command line argument or up to 10-process inputs otherwise.

Deliverables:

1. Software Design Document in a digital format, e.g. PDF, MS Word, etc.
2. C Implementation Code in a digital format, e.g. C file(s).

Include the following deliverables in an archive (zip) and submit it via the La Roche Blackboard. Please make sure that your Software Design Document along with all implementation files are marked with your name.

Acknowledgements:

This document is based in part on the work done by Professor Sunghee Sunny Kim at the Department of Computer Science in Gettysburg College, Gettysburg, PA.