**CSE 410** **Spring 2017**

# Computer Project #9

**Assignment Overview**

This assignment focuses on process management within an operating system, including processor scheduling. You will design and implement the C/C++ program which is described below.

It is worth 50 points (5 % of course grade) and must be completed no later than 11:59 PM on Thursday, 4/20.

**Assignment Deliverables**

The deliverables for this assignment are the following files:

**`proj09.makefile`** – the makefile which produces "proj09"
**`proj09.student.c`** – the source code file for your solution

Be sure to use the specified file names and to submit your files for grading via the CSE handin system before the project deadline.

**Assignment Specifications**

The program will simulate the steps that an operating system takes to manage user processes. The operating system uses the five-state process model described in the specifications for Project #8.

1. Your program will process a file which contains the simulation parameters and the process stream. The first four lines of the file will contain the simulation parameters (one item per line):

a) Number of CPUs in the system (an integer value; 1 for this assignment)
b) Number of PCBs in the system (an integer value; 1 or more for this assignment)
c) Length of the simulation in ticks (an integer; 0 or more for this assignment)
d) Short-term scheduling algorithm (a string; "FCFS" or "Priority" for this assignment)

2. Zero or more lines will follow the simulation parameters to represent the process stream. Each line will contain the following fields, separated by blanks:

a) Process identification number
b) Priority (from 1 to 3)
c) Number of CPU bursts
d) Burst time (in ticks)
e) Blocked time (in ticks)
f) Arrival time (in ticks since start of simulation)

The number of CPU bursts is an integer value representing the number of times that the process needs access to the CPU.

The burst time is an integer value representing the length of time that the process uses the CPU before issuing a service request.

The blocked time is an integer value representing the length of time that the process is blocked after issuing a service request.

3. On each tick of simulated time, the program will do zero or more of the following (in order):

a) Recognize a request issued by a process in the Running state
b) Recognize when a process in the Blocked state has become unblocked
c) Recognize the arrival of a new process
d) Allocate a PCB to a process in the New state
e) Dispatch a process in the Ready state

4. When a process terminates, the program will display the following information about that process:

a) All process parameters (items (2a) to (2f) above)
b) Departure time (in ticks since start of simulation)
c) Cumulative time in the New state (in ticks)
d) Cumulative time in the Ready state (in ticks)
e) Cumulative time in the Running state (in ticks)
f) Cumulative time in the Blocked state (in ticks)
g) Turnaround time (in ticks)
h) Normalized turnaround time (with two fractional digits of accuracy)

5. The string "FCFS" represents non-preemptive first-come-first-served scheduling. The string "Priority" represents non-preemptive priority scheduling, where 1 represents the highest priority.

6. The program will accept two command-line arguments: an integer representing the value of N and the name of the input file.

7. The program will display the simulation parameters (items (1a) to (1d) above) before processing the process stream.

8. The program will display the following information at the end of the simulation:

a) The current tick
b) Process ID of each process in the New state
c) Process ID of each process in the Ready state
d) Process ID of each process in the Running state
e) Process ID of each process in the Blocked state

That display will reflect the state of the processes after all transitions associated with that tick have occurred, but before the tick has been incremented.

In addition, if the value of N (the first command-line argument) is a positive integer value, the program will display items (8a) to (8e) after every N ticks.

9. The program will include appropriate error-handling.

**Assignment Notes**

1. In Project #10, you will extend your program from Project #9, so you would be wise to develop your program in a modular fashion and to comment your source code.

2. For this assignment, you may assume that contents of the input file will be valid, and that there are one or more processes in the process stream.