CS2106 Operating Systems
Semester 1 2019/2020

Week of 9th September 2019
Tutorial 3
Process Scheduling

1. (Putting it together) Take a look at the given mysterious program **`Behaviors.c`**. This program takes in one integer command line argument **D**, which is used as a **delay** to control the amount of computation work done in the program. For the part (a) and (b), use ideas you have learned from **Lecture 3: Process Scheduling** to explain the program behavior.

   Use the command taskset --cpu-list 0 ./Behaviors D
   This restricts the process to run on only one core.
   Warning: you may not have the `taskset` command on your Linux system. If so, install the `util-linux` package using your package manager (apt, yum, etc).

   Note: Refrain from using Windows Subsystem for Linux for this question as it does not produce the expected results.

   a. **D** = 1.

   b. **D** = 100,000,000 (note: don't type in the ",")

   c. Now, find the **smallest D** that gives you the following interleaving output pattern:

Interleaving Output Pattern
[6183]: Step 0 **[6184]: Step 0** [6183]: Step 1 [6184]: Step 1 **[6183]: Step 2** **[6184]: Step 2** [6183]: Step 3 [6184]: Step 3 **[6183]: Step 4** **[6184]: Step 4** [6184] Child Done! [6183] Parent Done!

   What do you think "**D**" represents?

   *Note: "**D**" is machine dependent, you may get very different values from your friends!.*

2. (Walking through Scheduling Algorithms) Consider the following execution scenario:

Program A, Arrives at time 0
Behavior (C**X** = CPU for **X** Time Units, IO**X** = I/O for **X** Time Units): C**3**, IO**1**, C**3**, IO**1**

Program B, Arrives at **time 0**
Behavior: C**1**, IO**2**, C**1**, IO**2** C**2**, IO**1**

Program C, Arrives at **time 3**
Behavior: C**2**

a) Show the scheduling time chart with First-Come-First-Serve algorithm. If two tasks arrive at same time, pick them in alphabetical order. For simplicity, we assume all tasks block on the same I/O resource.

Below is a sample sketch up to time 2:

b) What are the turnaround times and the waiting times for program A, B and C? In this case, waiting time includes all time units where the program is ready for execution but could not get the CPU.

c) Use **Round Robin** algorithm to schedule the same set of tasks. Assume time quantum of **2 time units**.

d) What is the response time for tasks A, B and C? In this case, we define response time as the time difference between the arrival time and the first time when the task receives CPU time.

3. (Adapted from Midterm 1516/S1 – Understanding of Scheduler)

a) Give the **pseudocode** for the **RR scheduler function.** For simplicity, you can assume that all tasks are CPU intensive that run forever (i.e. there is no need to consider the cases where the task blocks / give up CPU). Note that this function is invoked by timer interrupt that triggers once every time unit.

Please use the following variables and function in your pseudocode.

Variable / Data type declarations
Process **PCB** contains: **{ PID, TQLeft, … }** // TQ = Time Quantum, other PCB info irrelevant. **RunningTask** is the PCB of the current task running on the CPU. **TempTask** is an empty PCB, provided to facilitate context switching. **ReadyQ** is a FIFO queue of PCBs which supports standard operations like **isEmpty()**, **enqueue()** and **dequeue()**. **TimeQuantum** is the predefined time quantum given to a running task.
"Pseudo" Function declarations - black boxes that you can use
`SwitchContext( PCBout, PCBin );` Save the context of the running task in ***PCBout***, then setup the new running environment with the PCB of ***PCBin***, i.e. vacating ***PCBout*** and preparing for ***PCBin*** to run on the CPU.

b) Discuss how do you handle blocking of process on I/O or any other events. Key point: Should the code in (a) be modified (if so, how)? Or the handling should be performed somewhere else (if so, where)?

4. (Predicting CPU time) In the lecture, the ***exponential average*** technique is briefly discussed as a way to estimate the CPU time usage for a process. Let us try to see this technique in action. Use **$\text{Predicted}_0 = 10$ TUs** and **$\alpha = 0.5$**. Predicted_0 is the estimate used when a process is first admitted. All subsequent predictions use the formula:

$$\textbf{Predict}_{N+1} = \alpha\textbf{Actual}_N + (1-\alpha)\textbf{Predict}_N$$

Calculate the error percentage ($|\textbf{Actual} - \textbf{Predict}| \ / \ \textbf{Actual} * 100\%$) to gauge the effectiveness of this simple technique. CPU time usage of two processes are given below, fill in the table as described and explain the differences in error percentage observed.

Process A			
Sequence	**Predicted**	**Actual**	**Percentage Error**
1	**10**	9	11.1%
2		8	
3		8	
4		7	
5		6	
		Average Error:	

Process B			
Sequence	**Predicted**	**Actual**	**Percentage Error**
1	**10**	8	25%
2		14	
3		3	
4		18	
5		2	
		Average Error:	

Additional Questions (discussed if time allows):

Review Tutorial 2, Question 3