**PA2 : OS Threads Simulation**

Please write a Java program to simulate the management of OS processes and threads in the system.

**PCT** – You need to create a process-control table (**PCT**) of 16 rows (to manage up to 16 process-threads) and 6 columns. The 6 columns of PCT are:

1. process id: such as 1234, 4567, and 8901. // 4 digits integer

2. **thread id**: such as 1, 2, … up to 16. // 2 digits integer // ***This is a new column for PA2.***

3. status: such as running, waiting, ready, terminated. // String

4. cpu time used so far: such as 0, 1, 2, 3, …. The initial value is 0. // 2 digits integer

5. job priority: 1, 2, 3, 4 OR top, high, medium, low // 1 digit integer or String

6. current instruction counter: such as 1, 2, 3, …, 12. The initial value is 0. // 2 digits integer

**PIT** – You need to create a process-instruction table (**PIT**) of 16 rows (to manage up to 16 process-**threads**) and 12 columns. Each column shows the instruction to be executed (by the OS). The instruction can be compute-1, computer-2, compute-3, input-1, input-2, output-1, output-2, or **finish**. Each process can have up to 12 instructions to be executed with the last instruction being “finish” normally.

**CPUtimer** – You need to create a global (i.e., **static** in Java) variable called **CPUtimer**, which will track the execution of each process-**thread** to run through its instructions from PIT column 1 until column 12 or “finish” is encountered (whichever comes first).

**numCPUs** – You need to create a global (i.e., static in Java) variable called **numCPUs**, which specifies the number of processors that are operational in this system. Its value can be 1, 2, 3, 4, 5, 6, 7, or 8. The initial default value is 4.

**Data** – You need to preload the data into the **PCT** table, which is a String array of 17 by 7 (and don’t use row 0 and column 0 for your program’s readability and maintainability). You also need to preload the data into the **PIT** table, which is a String array of 17 by 13 (and don’t use row 0 and column 0 for your program’s readability and maintainability).

**RUN** command – When user issues “**run**” command, your system will execute the next instruction of each process-**thread** of top (then high, then medium, then low) job priority. You can only execute K process-threads for each run command where K is the number of processors as set in the **NumberOfCPUs** variable. In other words, you need to find the next K process-threads based on their job priorities. You need to keep track of the current instruction counter for each process-**thread**. For the first “run”, you will execute the first instruction of each process-thread. And so on until all process-threads are terminated. After the “**finish**” instruction is executed, you need to change the status of this process-thread to “terminated”, and you should not do anything for this process-thread afterward. After the instruction 12 (which is the last instruction) is executed for a process-thread, that process-thread must be terminated.

Your program must allow user to enter any of the following commands:

1. **run** -- “run” command is asking your program to execute all active processes one CPUtime. You must show what you have run for all process-**threads**. For the first run, your output may be like “Run process 1 *thread* 1 instruction 1; Run process 2 *thread* 1 instruction 1; …”. After a few runs, your output may be “Run process 5 *thread* 2 instruction 4;”. Finally, your output for run command may be “There is no instruction to run for any process-*thread*!”.

2. **show pct –** To show the complete contents of the process control table (PCT).

3. **show pit –** To show the complete contents of the process instruction table (PIT).

4. **ps** – To show the complete contents of PCT for only ***active*** processes. A process is ***active*** if any of its threads is not terminated yet.

5. **help** – To list all the available system commands with brief explanations.

6. **kill** *pid* – To kill and terminate the process *pid* and all its threads at the same time.

7. **set** **numCPUs** *K* – To set the system variable numCPUs to value *K*.

8. **set** **OS** N – To load PCT and PIT of OS N where N is a number for the set of PCT and PIT data. You may save 3 sets of PCT and PIT data in your program. The command set OS 1 will load the first set of data. The command set OS 2 will load the second set of data. The command set OS 3 will load the third set of data. This will allow you to run your system 3 times with 3 different sets of data, and you don’t have to recompile your program.

**Testing** – You must test your program **3** times with different PCT and PIT contents for each test. For each test, you may issue commands in this order for example: help; show pct; show pit; run; run; ps; run; show pct; show pit; run; run; run; show pct; show pit; ps; … ; kill 1234; set numberofcpus 8; set OS 2; help; show pct; show pit; run; show pct; run; show pct; … ; set OS 3; show pct; show pit; run; show pct; run; show pct; …

=================================================================================================================.

**How to submit your Lab or Project Assignment (PA)?**

(1) Each program must be well-documented with block comments and proper line comments. The

beginning of each program must have a block comment to show your name, date, and purpose.

The following is an example of block and line comments.

// Author: Dr. Simon Lin

// Date: 2/17/2016

// Purpose: To perform OS threads simulation

int pid , tid ; // process id , thread id

(2) You must submit the following items as attachments through sakai.apu.edu.

(a) All source programs (i.e., all **.java** files), and

(b) One WORD document (i.e., **.doc** or **.docx** file) containing all source programs’ listing and the

output of at least **3** test runs. You may use snapshots or copy-and-paste method to insert those

input/output onto your WORD document.

============================================================================================================================.

Grading Rubric:

You got \_\_\_ points out of 100 for **CS250 PA2**. Thank you for your excellent/good work.

[ ] -10 points for each day late.

[ ] 30 points – Your program must be fully tested at least 3 times with their complete output shown on your Word document.

[ ] 10 points – Your program must be well-documented.

[ ] -5 points if your program did not have block comment to show your name, date, and purpose.

[ ] 30 points – You must follow the program specification to develop your programs properly and completely.

[ ] 20 points – You must submit your WORD document.

[ ] 5 points – Your WORD document must show “**CS250 PA2**” and your full name on the **header**.

[ ] 5 points – Your WORD document must be page-numbered on the footer with format “Page 2 of 15” (for example).

============================================================================================================================.

Please use this Word document as a template for your Word document to be submitted. Please delete everything above.

Please copy all your source programs into here:

Please copy all your testing output into here: