LE/EECS 3221 – Operating System Fundamentals

Fall 2019

Programming Assignment 3

Submission Deadline: December 01, 2019 before 23:59

Objectives

- Learn the process and thread life cycle
- Learn how to use system calls related to threads
- Learn how multi level scheduling works

Submission Requirements

Please submit your results/output in the solution template provided.

Please submit your source code files separately (.c or .cpp etc.).

The submission deadline is December 01, 2019 before 23:59.

Assignment Requirements and Setup

You are required to perform the tasks in a Linux environment using C/C++ language. Before starting the tasks, first change your command prompt as discussed earlier. As a result, make sure that you see your Student ID in command prompt every time you write a new Linux command.

Question 1: Simulating a Multi Level Scheduler

Write a C/C++ program that will simulate the operation of a kernel with two-level-scheduler (short term and long term) using the Pthread library. The main program (kernel) will have two threads, each representing one of the two schedulers (implemented in code as two separate functions). There will be two Queues (use the Linux data structures from chapter 1 or create your own queue classes/structures): ready queue and job queue. Also, implement a struct with two elements: PID (unique ID of process) and Time (total execution time of process).

Step 1: Populate the job Queue with 100 processes. For each process, auto increment (starting from 1) the PID and randomly select Time (in the range 1-30). Display logging message to the screen: "[Kernel] Process X created with Time = Y"

Step 2: Create the two threads for the two schedulers.

Step 3: Invoke the long-term scheduler. Every time long-term scheduler method is invoked, display the logging message: "[Kernel] Long Term Scheduler Invoked" from the main program. First, it will display a logging message to show the current content of the job queue and ready queue as:

"[LTS] Job Queue: [Process X1: Time Y1], …."

"[LTS] Ready Queue: [Process X2: Time Y2], …." or "[LTS] Ready Queue: EMPTY"

Now, Long-term scheduler method will dequeue an element (process) from the job queue and enqueue it into the ready queue. Display logging message to the screen: "[LTS] Process X removed from the Job Queue and inserted to the Ready Queue". However, the ready queue has maximum length 5; if the ready queue is full, display logging message to the screen: "[LTS] Ready Queue is Full, cannot enter more". Long term scheduler will try to insert multiple processes to the queue if there are free spots. Now, Long-term scheduler will again display the content of the two queues using same format as mentioned above and will pass the control to short-term scheduler.

Step 4: Display the logging message: "[Kernel] Short Term Scheduler Invoked" from the main program; this message will be displayed every time the short term scheduler is invoked. Then, short term scheduler will display the content of two queues using the format discussed earlier, change LTS to STS. Then, short-term scheduler method will dequeue an element (process) from the ready queue and display the logging message to the screen: "[STS] Process X now executing". It will then reduce its time by two and enqueue it at the end of the ready queue. Also display the logging message to the screen: "[STS] Process X with remaining time Y enqueued to the Ready Queue". If the message has consumed its entire time, then it will not be enqueued to the ready queue and the logging message will be: "[STS] Process X terminated". Short term scheduler will repeat this action five times i.e. serve five processes. Display the content of the two queues once again and then pass the control to the long-term scheduler.

Step 3 and 4 will be repeated until all the elements (processes) are finished.

- **#1-A:** Submit your source code in a separate yourid.c/yourid.cpp file.
- **#1-B:** Run your program and redirect the output to a text file; name it output.txt. Submit this file.