LE/EECS 3221 – Operating System Fundamentals

Winter 2019

Programming Assignment 2

Submission Deadline: March 3, 2019 before 23:59

Objectives

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

Submission Requirements

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

The questions that involve performing task on the computer, your answer will be the screen shot of your steps; writing the answer in the text form is not acceptable.

The questions that involve coding/script, you have to write your code/script in text form in the document. Also, separately upload the source code (.c etc.) files.

The submission deadline is March 3, 2019 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: Creating Your Own Command Shell

Write a program that will display a prompt to the user as following where the user can enter their commands.

```
--SHELL--YOURID>>
```

On this prompt it will accept regular Linux commands from the user e.g.

```
--SHELL--YOURID>> ls -l
```

This command then will be executed by creating a child process and running the command in that child process. However, the output will not be displayed immediately to the user; rather, will be stored in a named pipe. Once, the output is ready (child process will indicate to the parent process which is your shell (you are free to pick your logic here)), the parent process will indicate the user that the output of the process is ready, and if user agrees, it will be displayed on the screen.

The user will have two choices to run a command: foreground and background. In case of foreground request, when user will submit a command, the prompt will not be returned to the user. It will be returned only when the output is ready, as mentioned above. Below is an example output. The "FG" prefix indicates that command should be executed in foreground:

```
--SHELL--YOURID>> FG ls -l
….working on request…..
….output is ready. Display it now [Y/N]>>Y
Response of command
--SHELL-YOURID>>
```

In case of background request, when user will submit a command, a message indicating the submission of the command will be displayed and the prompt will be returned to the user for further commands. When the output is ready, the user will be indicated of that and if agreed the output will be delayed. Below is an example output. The "BG" prefix indicates that command should be executed in foreground:

```
--SHELL--YOURID>> BG ls -l
….request submitted, returning prompt…..
--SHELL-YOURID>>
….output for "BG ls -l" is ready. Display it now [Y/N]>>Y
Response of command
--SHELL-YOURID>>
```

The key requirements in this program are: running the request in child process and implementing the FG/BG feature. Additional algorithmic details are not significant, and students are free to pick their logic.

- **#1-A:** Write/type your source code in your submission file/document. Also, submit the .c file separately.
- **#1-B:** In your submission file/document, add a screenshot which displays:
    - the execution of your program with the FG case
    - the execution of your program with the BG case

Question 2: Simulating a Multi Level Scheduler

Write a 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): ready queue and new queue. Also, implement a struct with two elements: PID (unique ID of process) and Time (total execution time of process).

At the start of the program, populate the new Queue with 100 processes. For each process, auto increment (staring from 1) the PID and randomly select Time (in the range 1-30). Then create the two threads for the two schedulers. Now invoke the long-term scheduler.

Long-term scheduler method will dequeue an element (process) from the new queue and enqueu into the ready queue. However, the ready queue has maximum length 5. In this method, display suitable logging message to the screen i.e. start, action, stop. Long-term scheduler will pass the control to short-term scheduler.

The short-term scheduler method will dequeue an element (process) from the ready queue, reduce its time by two, enqueu it to the ready queue. It will repeat this process five times and then pass the control to the long-term scheduler. If the Time of a certain element (process) reaches zero then it will be permanently removed from the ready queue. Also, in this method, display suitable logging message to the screen i.e. start, action, stop.

The program will end when all the elements have zero Time left.

- **#2-A:** Write/type your source code in your submission file/document. Also, submit the .c file separately.
- **#1-B:** In your submission file/document, add a screenshot which displays the execution of your program