Operating Systems Laboratory (CS39002)
Spring Semester 2018-2019

Assignment 4: Implement a CPU scheduler on top of Linux using threads and compare scheduling algorithms

Assignment given on: February 4, 2019

Assignment deadline: February 11, 2019, 2:00 PM

Implement a virtual CPU scheduler on top of the Linux kernel using POSIX threads, and evaluate CPU scheduling algorithms by running a synthetic job mix on the scheduler. The details of the specification are as follows.

a) Create *N* concurrent threads that will be scheduled by the virtual scheduler. Each thread will be either a ***producer (P)*** or a ***consumer (C)***, selected with equal probability. Each of the P threads will be generating 1000 pseudo-random integers, and storing them in a ***shared BUFFER*** of maximum capacity *M*; if the buffer is full, the thread will wait. Each of the C threads will be repeatedly removing an element from the BUFFER; if the buffer is empty, the thread will wait. The P and C threads are referred to as ***WORKER*** threads.

b) Each of the WORKER threads will have ***signal handlers*** installed for handling the user-defined signals SIGUSR1 and SIGUSR2. SIGUSR1 will be used to put the thread to sleep, while SIGUSR2 will be used to wake up the thread to resume execution. The running status of all the threads will be stored in a shared data structure called ***STATUS***.

c) Another thread, called the ***SCHEDULER*** thread, will be created that will be sending sleep/wakeup signals to the WORKER threads. It will implement a round-robin scheduling algorithm with a specified time quantum (say, 1 second). It will run one of the *N* WORKER threads at a time, while the other *N-1* threads will be put to sleep. During context switch, the currently running thread will be put to sleep, while the next thread in the READY queue will be activated.

d) Another thread will be created, called ***REPORTER*** thread, which will continuously monitor the ***STATUS*** data structure, and display relevant messages on the screen whenever a context switch or thread termination takes place. It will also display the number of elements in BUFFER.

Hint: *Use the POSIX Pthread library for creating/managing the threads.*

Submission Guideline:

- Create the program as a single file as **`Ass4_<groupno>.c`** or **`.cpp`**, and upload it.

Evaluation Guidelines:

While entering marks, the partwise break up should also be entered according to the marking guidelines given below. There is a separate component for individual assessment, based on how the student answers questions.

Sl	Items	Marks
(a)	Creation of concurrent threads	5
(b)	Storing data in a shared buffer	5
(c)	Correctly implementation of signal handlers	8
(d)	Correctly synchronised accesses to buffers	8
(e)	Implementation of scheduling policy	8
(f)	Correctly synchronised accesses to status for updating and reporting	10
(g)	Overall correctness	6
	Total	50