Operating Systems - Monsoon 2020

Arani Bhattacharya, Sambuddho Chakravarty

November 20, 2020

Assignment 4 (Total points: 60)

Due date: Dec. 4, 2020. Time: 23:59 Hrs.

The pthread library provides the `pthread_mutex` structure to enable easy mutual exclusion. Design a counting semaphore structure using mutex called `my_semaphore` so that multiple pthreads can synchronize access to some limited/shared resource. Note that the counting semaphore should use the signalling mechanism the way semaphores work, *i.e.* signalling a thread that waits on a semaphore. Your counting semaphore should implement the primitives `wait()`, `signal()` and `signal(printValue)` for debugging.

Use the semaphore you have so defined to solve the following modified version of dining philosopher problem. Consider the scenario where you have a similar case of *k* philosophers and *k* forks. But in this case, the dinning table also has a pair of sauce bowls, both of which are needed simultaneously for a philosopher to eat. Write a program that simulates the philosophers using threads, and the resources (forks and sauce bowls) using a shared variable. Your simulation should print the following message on the console whenever a philosopher gets to eat: "Philosopher $< thread\_id >$ eats using forks $< fork\_no\_x >$ and $< fork\_no\_y >$. The simulation keeps running in an infinite loop. Then, use `my_semaphore` to avoid any possible deadlocks. You may also implement non-blocking variants of `wait()` and `signal()` using the non-blocking functions available in the Pthreads library.

What To Submit

- The C program sources with proper comments describing each section of your code. Inside each file your name and roll number must be present as a comment on top. Naming convention for the codefiles is $< filename >$ *_RollNo.c*
- `Makefile` to compile the source and generate the running binary.
- Short write-up (of up to 1 page) describing the functionality and technique used in the program. Submit the document in PDF only, using the naming convention *WriteUp_RollNo.pdf*
- Submit all the above in a zipped folder with the naming convention *RollNo.zip*.

Grading Rubric

- Successful compilation using Makefile – 5 points.
- Implementing the synchronization primitives, *viz.* `signal()` and `wait()` (both blocking and non-blocking variants) using the Pthread library. – 25 points
- Successfully demonstrating the working solution of the dinning philosophers' problem with the primitives that clearly shows that the deadlocks don't occur. – 25 points
- Writeup – 5 points.

Late Submission Policy

- Submitted on or before December 4, 2020 (23:59 hrs) – No points deducted.
- Submitted after December 4, 2020 (23:59 hrs) – 0 points or students can use the one-time late submission policy if still available with them.