COMP 2240 – Operating System

Review

Definition:

Semaphore:

A semaphore controls access to a shared resource through the use of a counter. If the counter is greater than zero, then access is allowed. If it is zero, then access is denied.

Deadlock:

It is a state in which each member of a group is waiting for some other member to take action. It is a common problem in multiprocessing systems, parallel computing, and distributed systems, where software and hardware locks are used to handle shared resources and implement process synchronization.

Starvation:

In starvation, threads are also waiting for each other. But here waiting time is not infinite after some interval of time, such as Thread.sleep();

Waiting thread always gets the resources whatever is required to execute thread run().

How to test the program:

I used IntelliJ as my IDE, so there is a function called “Debug”, I can run the program by using the debug function, and see the result line by line, then I can test the deadlock and semaphore.

For mutual exclusion, suppose two or more processes require access to a single nonshared resources. In task 3, class “CClient” and “HClient” share the same “CoffeeShop” and   
“Main”. I call the “CClient” when I read “C” in the file, and call “HClient” when I read “H” in line.

For deadlock, I created Thread. In task I put the process into sleep for a certain time and continues. In task 2 and 3, I use semaphore release and semaphore acquire in the code.

For starvation free, I made sure there is no releaseAll(); in my code. So, there will not have a starvation.

In my coding, I also make sure I acquired the semaphore and released it.