Jalen Powell

October 28, 2022

COMP 3500

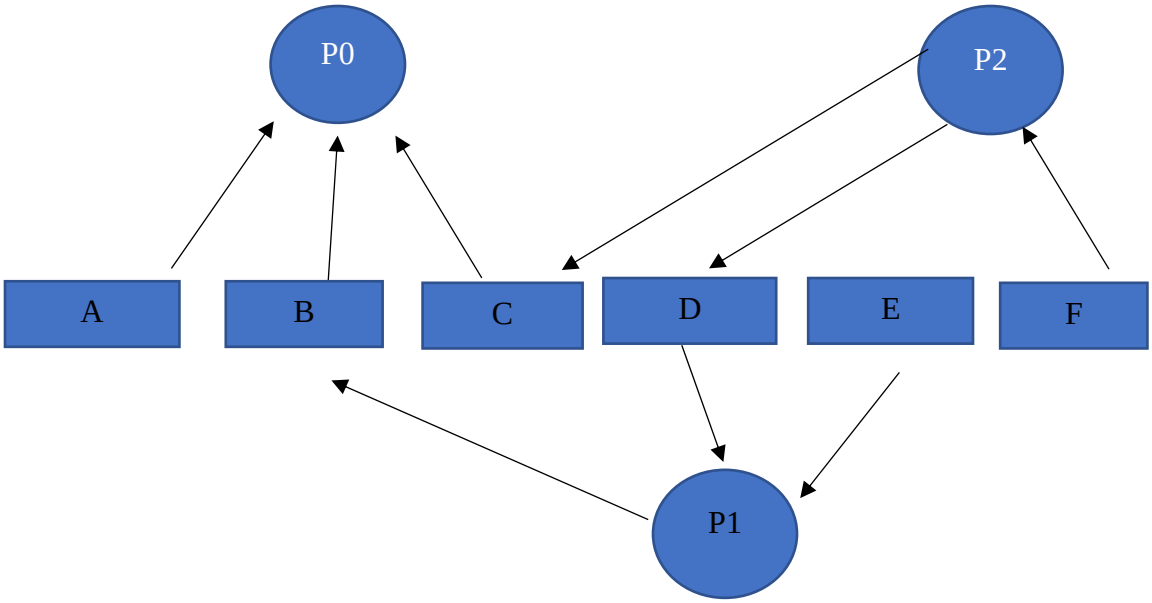
Project 4

Question 1:

a.) The possibility of a deadlock occurring in this implementation is high as many units of one type is needed to execute the task. There could be three processes fighting for two resources of type A and just one process fighting for a resource type B. If the three processes holding resources on type A continue, the one process vying for type B will be halted. This will cause a resource stalemate to happen, and the system will be unable to advance.

b.) Modifying the order to avoid a deadlock would look like this:

- P1 wants to use resource B while it is being held by the process P0. P1 can only execute when it has the resources B, D, and E. Resource B will not be given to P1 until P0 releases it, so the only way to prevent the deadlock from happening is to put a get request of resource B for P1 after B is released by P0.



Question 2:

Yes, the processes **Foo** and **Bar** can be blocked forever and here is how.

- **Foo()** executes "semwait(**S**)". After this line, the value of **S** will be 0.
- **Bar()** executes "semwait(R)". After this line, the value of **R** will be 0.

Now both processes will be blocked forever. Process **Foo()** will block on "semwait(**R**)" since the value of **R** is 0. **Foo()** must wait until **R** to become 1. The same steps apply to **Bar()**. This process will block on "semwait(**S**)" and must wait until **S** becomes 1. This event will never complete.

Question 3:

Deadlock detection is to detect situations when the application is blocked by two conflicting resources and then tries to prevent those situations from happening. Deadlock avoidance is more like detection as they both rely on detecting the possible deadlock to ensure the application never gets stuck. Deadlock prevention detects the situation instantly, although it is still difficult to prevent deadlocks. Adding a deadlock handler to the code would suffice, but that can be expensive.

Question 4:

Yes, the system is deadlock-free. If the system is deadlocked, this would mean that each process is holding one resource and is in waiting state for one more. Since there are three processes and four resources, one process will have to hold two resources. This process wont need any more resources so it will return the resources once finished.