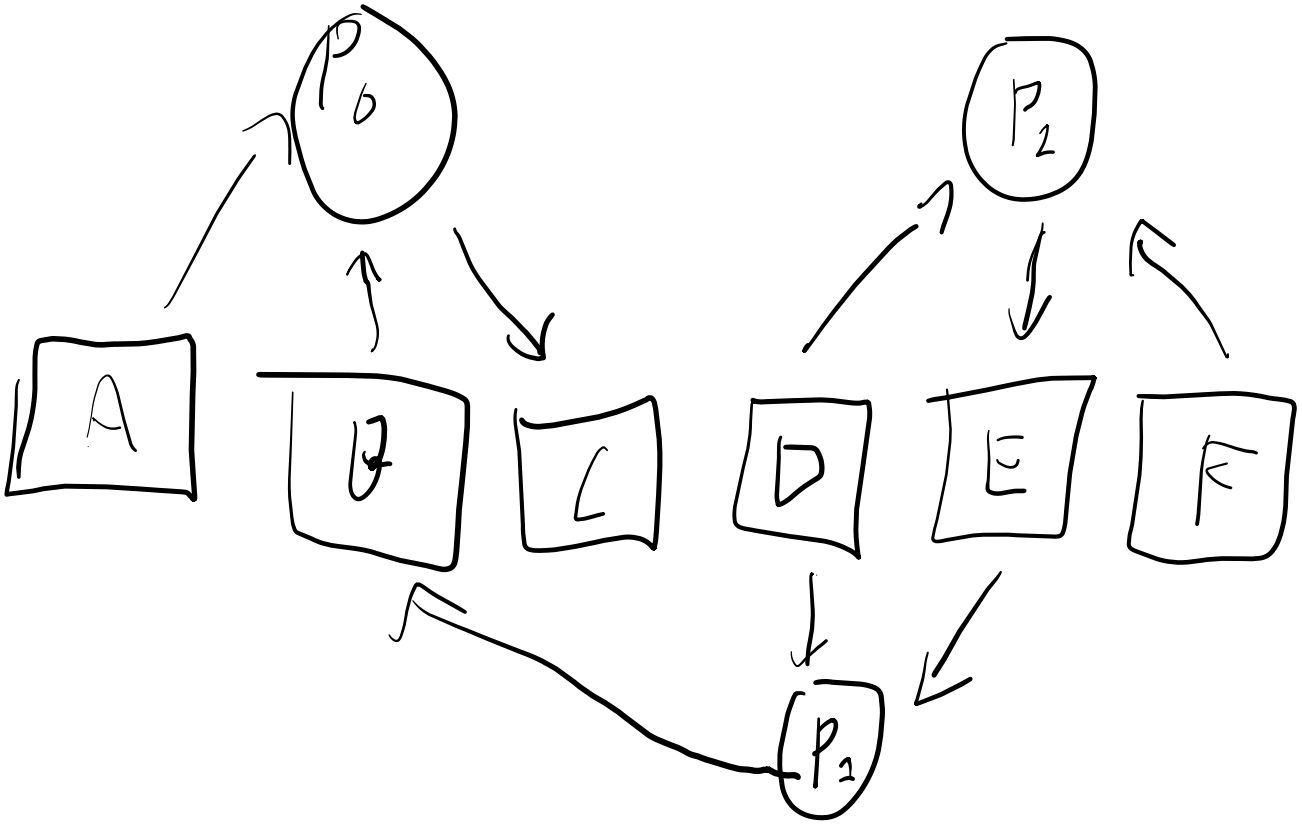
JP Halog jdh0102

1 A. P0 = A -> P1 = D -> P2 = C -> P0 = B -> P1 = E -> P2 = F ->.

Then these end up waiting P0 = C -> P1 = B -> P2 = D



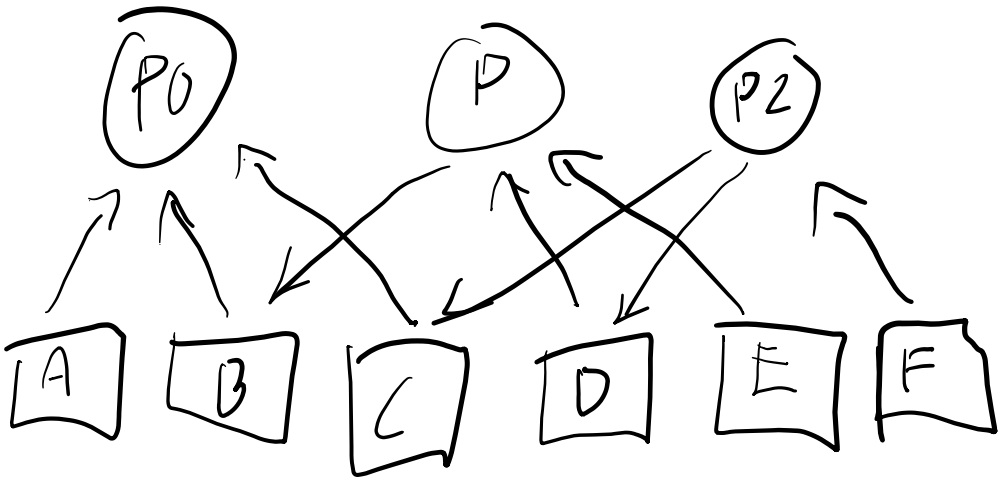

Loop/cycle : B -> $P_0$ -> C -> $P_3$ -> D -> $P_2$ -> B

cycle indicates deadlock

1 B. In Process P0 order by: get(B), get(C), then get(A)

In Process P1 order by: get(D), get(E), then get(B)

In Process P2 order by: get(F), get(D), then get (C)



2. Yes, the concurrent execution results in both being blocked forever.

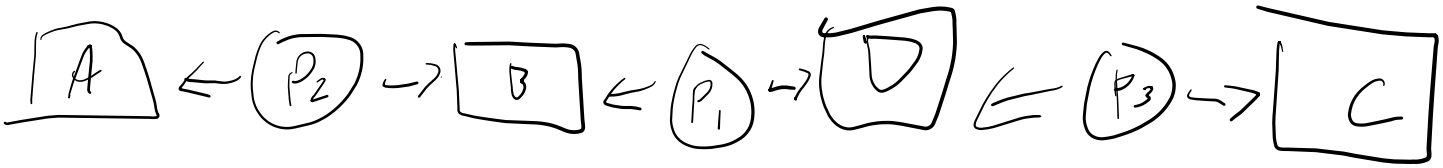
1. Foo() execute semwait(s) and value of S = 0
2. Bar() executes semwait(R) and value R = 0
   After this execution both processes are blocked forever since process foo() will be blocked on semwait(R) since R=0 and Process bar() will be blocked on semwait(S) ince value of S = 0 and will have to wait for S = 1.
   Since both are waiting on each other, the process cannot complete.

3. Deadlock avoidance prevents deadlocks by carefully allocating resources to processes and analyzes potential allocation scenarios to ensure system remains in a safe state. If a resource allocation request possibly leads to deadlock, it is denied until safe state can be guaranteed.

Deadlock detection examines system's resource allocation graph to identify the presence of a deadlock. Once a deadlock is detected, the system can kill the processes involved in the deadlock or roll back their actions to prevent the deadlock.

Deadlock prevention ensures that at least one of the necessary conditions for deadlock cannot occur. This can be applied through mutual exclusion, hold and wait, no preemption, or circular wait.

4.




Since there is no loop, no deadlock