ASP ASSIGNMENT 6

Kshitij Raj | UFID: 13584965

DEADLOCKS

Deadlock 1: userapp attempts to open device in Mode 1 twice.

- Device is set to mode1
- userapp opens the device for the first time and succeeds if there are no other open devices
- userapp opens the device for the second time without close it and results in a deadlock since it is trying to acquire sem2 which it already owns
- Since userapp is a single threaded process, there is no way it can execute further
- The file deadlock1.c implements a simple userapp which enters deadlock by trying to open the file twice.
- userapp should execute fine in mode2
- Gets stuck on line 49.

Deadlock 2: Two separate userapp threads attempt to change the Mode 2->1 at the same time.

- Userapp sets mode =2 initially
- It spawns two threads and both of them open the device
- Then, both the threads try to change mode to 1
- They end up in a deadlock since devc->count2 >1 holds true and both the threads give up.
- The file deadlock2.c implements the userapp for the above-mentioned behavior.
- Stuck on line 154 & 154.
-

Deadlock 3: Thrread 1 attempts to change the mode 2->1 while the second thread is opening the file.

- Thread1 has opened the file.
- Thread 2 tries opening the file but stalls since it is not able to acquire sem2.
- Thread 1 tries to change mode to 2 but since devc->count1 > 1, it goes into wait queue holding sem2.
- Thread 2 is waiting for sem2 and there is a deadlock.
- The file deadlock3.c implements this scenario
- Stuck on line 49 & 154

Deadlock 4: 2 threads (same process) calling ioctl

- Two threads are created which try to open the device
- Device is in mode1 and only one threads succeeds in acquiring sem2
- Operation is changed to mode2
- The second thread tries to switch mode back to mode1 and this causes a deadlock since count>1
- The file deadlock4.c implements this scenario