104062203 陳涵宇

1. Problem Set

1. The first known correct software solution to the critical-section problem for two processes was developed by Dekker. The two processes, P0 and P1, share the following variables:

```
        boolean flag[2]; /* initially false */
        int turn;
```

The structure of process Pi (i == 0 or 1) is shown in Figure 6.21. The other process is Pj (j == 1 or 0). Prove that the algorithm satisfies all three requirements for the critical-section problem.

Mutual exclusion:

Assume flag[0]==flag[1]==true. Since turn will only be either 0 or 1, P0 and P1 can't in critical section at the same time.

Bounded wait:

After Pi executes, it will set its flag to false and turn = j, so Pj can run.

Progress:

After one of the process is executed, it will set turn to the other process, thus to prevent the other process from indefinitely waiting.

2. Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in user-level programs.

If a user-level program can disable interrupts, then it can prevent from context switching and thus occupies resources.

3. How does the signal() operation associated with monitors differ from the corresponding operation defined for semaphores?

The signal() operation associated with monitors won't be memorized if there are no waiting threads. If there is a new thread later, the new thread will still be blocked. For semaphore, since semaphore can work as a counter, the following threads can immediately start.

2.4 Show your typescript. Run your code multiple times. Does it show the same or different output? Why?

It does not show the same output, since the time of each thread to sleep is different.