

# OSu HW3

110590004 資工三 林奕廷

## Writing exercise

### 7.8

Since the spinlock should only be used when the waiting time is short, attempting to acquire a semaphore may make the process sleep, which will hang the spinlock for too long, increase the overhead for the spinlock, and obey the usage of a spinlock.

### 8.20

- (a): Is safe in any circumstance.
- (b): It Is safe only if the Available is larger than the Need of every process.
- (c): It is safe only if the Max + Allocated doesn't exceed the Available.
- (d): It is safe in any circumstance.
- (e): It is safe only when the new process' requiring resources is ample.
- (f): It is safe in any circumstance.

### 8.27

(a)

Available = (2, 2, 2, 3)

P4, A  $\Rightarrow$  (3, 2, 2, 4)

P0, A  $\Rightarrow$  (4, 4, 2, 6)

P1, A  $\Rightarrow$  (4, 5, 3, 8)

P2, A  $\Rightarrow$  (5, 7, 7, 8)

P3, A  $\Rightarrow$  (6, 9, 8, 9)

The process can be executed in the order (P4, P0, P1, P2, P3), and the state is safe.

(b)

Available = (4, 4, 1, 1)

P2, A  $\Rightarrow$  (5, 6, 5, 1)

P4, A  $\Rightarrow$  (6, 6, 5, 2)













P1, A  $\Rightarrow$  (6, 7, 6, 4)

P3, A  $\Rightarrow$  (7, 9, 6, 5)

P0, A  $\Rightarrow$  (9, 11, 6, 7)

The process can be executed in the order (P2, P4, P1, P3, P0), hence the state is safe.

## 8.30

```
mutex ,  /* normal mutex */,  /* the bridge's lock */
N = 0 // the count of North farmers on the bridge
S = 0 // the count of South farmers on the bridge
north_to_south() {
    wait();
    N++;
    /* first farmer ought to lock the bridge */
    if (N == 1)
        wait();
    signal();
    ...
    /* farmer crossing the bridge */
    .cross();
    ...
    wait();
    N--;
    /* last farmer ought to release the lock */
    if (N == 0)
        signal();
    signal();
}
south_to_north() {
    wait();
    S++;
```

```

/* first farmer ought to lock the bridge*/
if (S == 1)
    wait(🌉);
signal(🔑);
...
/* farmer crossing the bridge */
👨.cross();
...
wait(🔑);
S--;
/* last farmer ought to release the bridge's lock*/
if (S == 0)
    signal(🌉);
signal(🔑);
}

```

## 9.15

(a)

The contiguous memory has severe external fragmentation problems, due to the free memory being broken into non-contiguous blocks.

The paging has almost no external fragmentation since the page size is fixed and can be reused easily.

(b)

The contiguous memory has less internal fragmentation since the method can much easier to allocate a memory exactly matching memory needs.

The paging has internal fragmentation too, it will occur on the last page allocated due to the memory needs not perfectly matching multiples of the page size.

(c)

It would be hard to share code between processes with contiguous memory. Since each memory allocates the memory contiguously, and there are no native sharing mechanics. It may require other complex operations to achieve.

The paging mechanic can make code sharing much easier. Since the shared code can be easily put into a page, and map the page into multiple requiring process's page tables.

## 9.24

(a)

$$2^{32} \div 2^{13} = 2^{19} \text{ entries}$$

(b)

$$2^{30} \div 2^{13} = 2^{17} \text{ entries}$$