VE482 Homework 5

519370910084 Lan Wang

Ex 1

1. **A system has two processes and three identical resources. Each process needs a maximum of two resources. Can a deadlock occur? Explain.**

No. Suppose both processes need two resources. When each process has one resource, there is still a free resource that can be given to one of them. Then the process with two resources will execute normally and release the resources, which will make the other process finish waiting.

2. **A computer has six tape drives, with n processes competing for them. Each process may need two drives. For which values of n is the system deadlock free?**

$n = 5$

3. **A real-time system has four periodic events with periods of 50, 100, 200, and 250 msec each. Suppose the four events require 35, 20, 10, and x msec of CPU time, respectively. What is the largest value x for which the system is schedulable?**

We need $\frac{35}{50} + \frac{20}{100} + \frac{10}{200} + \frac{x}{250} \leq 1$, which means:

$$700 + 200 + 50 + 4x \leq 1000 \Rightarrow x \leq 12.5$$

So the largest x equals to 12.5

4. **Round-robin schedulers normally maintain a list of all runnable processes, with each process occurring exactly once in the list. What would happen if a process occurred more than once in the list? Would there be any reason for allowing this?**

The process will get multiple turns when passing through the list, which leads to higher percentage of CPU time. This would be used if we want the most important process to spend the most CPU.

5. **Can a measure of whether a process is likely to be CPU bound or I/O bound be detected by analyzing the source code. How to determine it at runtime?**

Yes. Generally speaking, if there are many I/O operations, the process is more likely to be I/O bound. Otherwise, if it is going to do lots of calculations, it tends to be CPU bound.

To determine it at runtime, use `top`, `iotop`, `iostat`, etc.

Ex 2

Assuming three resources consider the following snapshot of a system.

Process	Allocated	Maximum	Available
P_1	010	753	332
P_2	200	322	
P_3	302	902	
P_4	211	222	
P_5	002	433	

1. **Determine the content of the Request matrix.**

   $$\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

2. **Is the system in a safe state?**

   Yes. In the sequence of: P_2, P_4, P_5, P_1, P_3

3. **Can all the processes be completed without the system being in an unsafe state at any stage?**

   Yes. Just consider the sequence in last question.

Ex 5

1. **Explain how to get a read lock, and write the corresponding pseudocode.**

```
void read_lock() {
    down(count_lock);
    if(count==0){ // only first reader lock the database
        down(db_lock);
    }
    count++;
    up(count_lock);
}
void read_unlock() {
    down(count_lock);
    count--;
    if(count==0){ // only last reader release the lock
        up(db_lock);
    }
    up(count_lock);
}
```

2. **Describe what is happening if many readers request a lock.**

   The database keeps locked, so that no writer can access it.

3. **Explain how to implement this idea using another semaphore called read_lock.**

```
void write_lock() {
    down(read_lock); // following readers are not able to read
    down(db_lock);
}
```

```
void write_unlock() {
    up(db_lock);
    up(read_lock);
}
void read_lock() {
    down(read_lock); // check the permission to read
    up(read_lock); // enables other readers to read
    down(count_lock);
    if(count==0){ // only first reader lock the database
        down(db_lock);
    }
    count++;
    up(count_lock);
}
void read_unlock() {
    down(count_lock);
    count--;
    if(count==0){ // only last reader release the lock
        up(db_lock);
    }
    up(count_lock);
}
```

4. **Is this solution giving any unfair priority to the writer or the reader? Can the problem be considered as solved?**

   No. Readers and writers have same priority for `read_lock`.