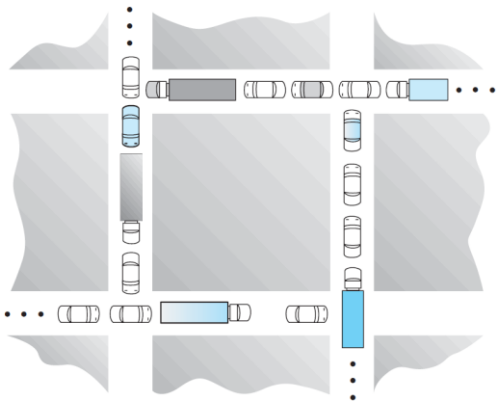
COSC1112/1114: Operating Systems Principles

Tutorial 06 (week 07)

1. **Consider the traffic deadlock depicted blow.**



   a) **Show that the four necessary conditions for deadlock indeed hold in this example.**
   b) **State a simple rule for avoiding deadlocks in this system.**

Answer:

   a) The four necessary conditions for a deadlock are (1) mutual exclusion; (2) hold-and-wait; (3) no preemption; and (4) circular wait. The mutual exclusion condition holds since only one car can occupy a space in the roadway. Hold-and-wait occurs where a car holds onto its place in the roadway while it waits to advance in the roadway. A car cannot be removed (i.e. preempted) from its position in the roadway. Lastly, there is indeed a circular wait as each car is waiting for a subsequent car to advance. The circular wait condition is also easily observed from the graphic.
   b) A simple rule that would avoid this traffic deadlock is that a car may not advance into an intersection if it is clear it will not be able immediately to clear the intersection.

2. **The program example shown below doesn't always lead to deadlock. Describe what role the CPU scheduler plays and how it can contribute to deadlock in this program.**

```
/* thread one runs in this function */
void *do work one(void *param)
{
        pthread mutex lock(&first mutex);
        pthread mutex lock(&second mutex);

        /* Do some work*/

        pthread mutex unlock(&second mutex);
        pthread mutex unlock(&first mutex);
        pthread exit(0);
}
```

```
/* thread two runs in this function */
void *do work two(void *param)
{
        pthread mutex lock(&second mutex);
        pthread mutex lock(&first mutex);

        /*Do some work*/

        pthread mutex unlock(&first mutex);
        pthread mutex unlock(&second mutex);
        pthread exit(0);
}
```

Answer:

If thread_one is scheduled before thread_two and thread_one is able to acquire bothmutex locks before thread_two is scheduled, deadlock will not occur. Deadlock can only occur if either thread_one or thread_two is able to acquire only one lock before the other thread acquires the second lock.

3. **Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free.**

Answer:

Suppose the system is deadlocked. This implies that each process is holding one resource and is waiting for one more. Since there are three processes and four resources, one process must be able to obtain two resources. This process requires no more resources and, therefore it will return its resources when done.