Lecture

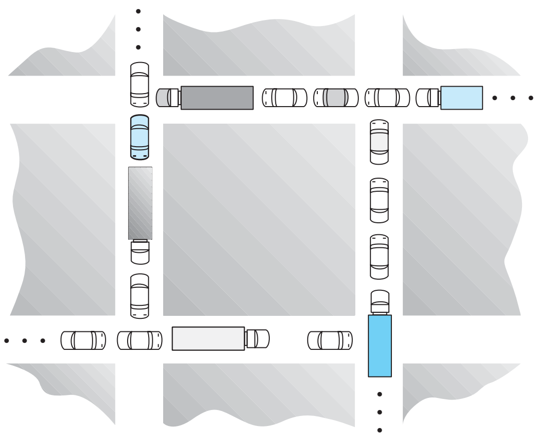
- In the lecture on April 07 we will mainly discuss Chapter 7 (Deadlocks) and start with Chapter 8 (Main Memory Management). In the lecture on April 08 we will mainly discuss Chapter 8. Paging will be described and dicussed in detail basd on a simulation tool.
- The 3rd mandatory assignment will be introduced on Tuesday or Wednesday.

Exercises

Note, as usual, that you find even more exercises including solutions here :
`http://codex.cs.yale.edu/avi/os-book/OS9/practice-exer-dir/index.html`

Prepare for the Tutorial Session on Thursday, April 09, 2015:
All exercises not discussed so far. In addition:

7.1 Consider the traffic deadlock depicted the Figure below (from the course book).



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.

7.2 Assume a multithreaded application uses only reader-writer locks for synchronization. Applying the four necessary conditions for deadlock, is deadlock still possible if multiple reader-writer locks are used?

7.4 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);
    }
```

7.5 Compare the circular-wait scheme with the various deadlock-avoidance schemes (like the banker's algorithm) with respect to the following issues:

   a. Runtime overheads

   b. System throughput

7.6 In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

   a. Increase Available (new resources added)

   b. Decrease Available (resource permanently removed from system)

   c. Increase Max for one process (the process needs or wants more resources than allowed).

   d. Decrease Max for one process (the process decides it does not need that many resources)

e. Increase the number of processes

f. Decrease the number of processes

7.7 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.

7.8 Consider a system consisting of m resources of the same type being shared by n processes. A process can request or release only one resource at a time. Show that the system is deadlock free if the following two conditions hold: i.) The maximum need of each process is between 1 and m resources, and ii.) The sum of all maximum needs is less than m + n

7.9 Consider the version of the dining-philosophers problem in which the chopsticks are placed at the center of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

7.10 Consider again the setting in the preceding question. Assume now that each philosopher requires three chopsticks to eat. Resource requests are still issued one at a time. Describe some simple rules for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.

7.12 Consider the following snapshot of a system:

	Allocation				Max			
	A	B	C	D	A	B	C	D
P_0	3	0	1	4	5	1	1	7
P_1	2	2	1	0	3	2	1	1
P_2	3	1	2	1	3	3	2	1
P_3	0	5	1	0	4	6	1	2
P_4	4	2	1	2	6	3	2	5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe.

a. Available = (0, 3, 0, 1)

b. Available = (1, 0, 0, 2)

7.13 Consider the following snapshot of a system:

	Allocation				Max			
	A	B	C	D	A	B	C	D
P_0	2	0	0	1	4	2	1	2
P_1	3	1	2	1	5	2	5	2
P_2	2	1	0	3	2	3	1	6
P_3	1	3	1	2	1	4	2	4
P_4	1	4	3	2	3	6	6	5

Assume the Available vector being (A,B,C,D)=(3,3,2,1). Answer the following questions using the banker's algorithm:

a. Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.

b. If a request from process P_1 arrives for (1, 1, 0, 0), can the request be granted immediately?

c. If a request from process P_4 arrives for (0, 0, 2, 0), can the request be granted immediately?

7.15 A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighboring town. The bridge can become deadlocked if both a northbound and a southbound farmer get on the bridge at the same time (Vermont farmers are stubborn and are unable to back up). Using semaphores, design an algorithm that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, and vice versa).

7.16 Modify your solution to Exercise 7.15 so that it is starvation-free.

7.11 We can obtain the banker's algorithm for a single resource type from the general banker's algorithm simply by reducing the dimensionality of the various arrays by 1. Show through an example that we cannot implement the multiple-resource-type banker's scheme by applying the single-resource-type scheme to each resource type individually.