CSE 430 Homework 5

Ryan Dougherty

Question 7.4

A possible method for preventing deadlocks is to have a single, higher- order resource that must be requested before any other resource...Compare this scheme with the circular-wait scheme of Section 7.4.4. Both this algorithm and the circular-wait scheme prevent the deadlock, so that requirement is satisfied. However, there is a difference between the two algorithms: the circular-wait algorithm is more efficient than this one because circular-wait allows the shared resources to be used by multiple processes (given no deadlock), whereas this algorithm only allows 1.

Another difference is that an implementation of this algorithm would be easier than circular-wait because for circular-wait, a process may not be able to predict what is the number of resources that it will use.

Question 7.11

Consider the traffic deadlock depicted in Figure 7.10.

(a) Show that the four necessary conditions for deadlock hold in this example.

(i) Mutual exclusion: the cross-streets are only used by only one line of cars.

(ii) Hold and wait: each line of cars is holding one cross-street (the resource), and is waiting for another to become available.

(iii) No preemption: the cross-streets are not freed (i.e. resource not released) until there are no cars left in it.

(iv) Circular wait: the west-moving line of cars is waiting for the south, which is waiting for east, which is waiting for north, which is waiting for west (i.e. there is a circle in waiting).

(b) State a simple rule for avoiding deadlocks in this system. There are many ways of avoiding deadlock (a complex solution would include using today's stoplights). A simple solution is for each car, determine if it can exit the intersection if it enters one. If yes, then proceed. Otherwise, wait until you can.

Question 7.23

Consider the following snapshot of a system: ... 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. The sequence $< P_0, P_3, P_1, P_2, P_4 >$ works. $Need_0 = (2, 2, 1, 1)$, so $Need_0 \leq Work = Available = (3, 3, 2, 1)$. Therefore, $Work = Work + Allocation_0 = (5, 3, 2, 2)$. We next pick P_3, and following the same algorithm gives a result for $Work = (6, 6, 3, 4)$. Since $Need_i$ for $i = 1, 2, 4 \leq (6, 6, 3, 4)$, we can pick any other sequence for the last three. We then finish the last three sequences, and move to Step 4 of the algorithm. Since $Finish_i == true$ for all i, the system is in a safe state.

(b) If a request from process P_1 arrives for (1, 1, 0, 0), can the request be granted immediately? Since $Request_1 \leq Need_1$, we go to Step 2. Since $Request_1 \leq Available$, we go to Step 3. We then follow the algorithm, which now has $Available = (2, 2, 2, 1), Allocation_1 = (4, 2, 2, 1), Need_1 = (1, 0, 3, 1)$, and the rest unchanged. We now check if the system is safe. Going through the algorithm, we can see that the sequence given in part (a) shows that the system is in a safe state. Therefore, the resources are allocated to P_1.

(c) If a request from process P_4 arrives for (0, 0, 2, 0), can the request be granted immediately? Since $Request_4 \leq Need_4$, we go to Step 2. Since $Request_4 \leq Available$, we go to Step 3. We then follow the algorithm, which now has $Available = (3, 3, 0, 1), Allocation_4 = (1, 4, 5, 2), Need_4 = (2, 2, 1, 3)$, and the rest unchanged. We now check if the system is safe. Going through the algorithm, we can see that for all processes, resource C has a value of 1 or greater, but what is available for C is 0. Therefore, the system is unsafe, and P_4 must wait (and the banker's algorithm says that the old resource-allocation state is restored).