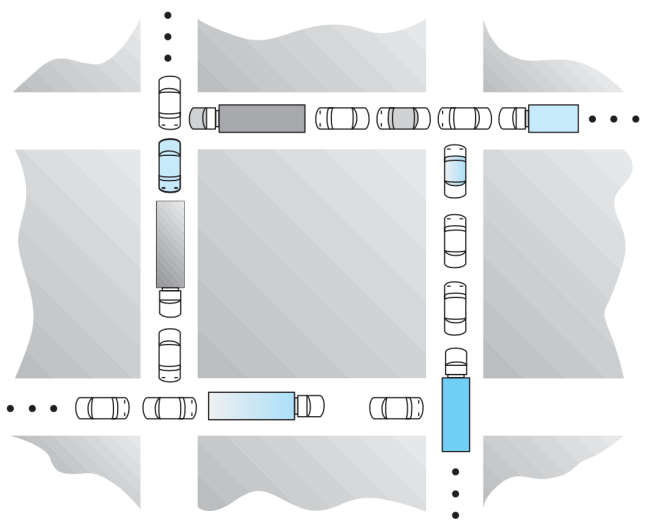
[hw#2] 7장 연습문제

7.11



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

Figure 7.10 Traffic deadlock for Exercise 7.11.

1) Mutual exclusion

최소 하나의 자원이 비공유모드로 점유되어야 한다. 현재 자동차가 한 라인으로 한 방향의 자동차 밖에 들어가지 못하므로 Mutual Exclusion 상황이다.

2) Hold and wait

하나의 자원을 점유한 채 다른 스레드에 점유된 자원을 대기해야한다. 현재 자동차가 모두 한 라인씩 점유를 하고 있으며, 다음 라인들을 기다리고 있으므로 Hold and wait 한 상황이다.

3) No preemption

자원들을 선점할 수 없어야 한다. 한번 들어온 자동차는 뺄 수 없으며, 앞서 traffic 이 해결 된 후에 앞으로 가서 빠지는 상황만 기다려야 한다.

4) Circular wait

대기하고있는 프로세스들이 꼬리에 꼬리를 물어야 한다. 현재 아래 방향으로 가는 라인이 오른쪽 진행방향을 기다리고, 위로 가는 방향을 기다리고, 왼쪽 방향을 기다리고 아래쪽 방향을 기다리며, Circular Wait 한 상황이다.

b. State a simple rule for avoiding deadlocks in this system.

각 방향마다 신호등을 설치하여 간단히 해결 할 수 있다. 교차로가 critical section 이므로, 우선 교차로에 진입하기 전, 위 아래로 신호등 각각 한 개씩과 좌우로 가는 방향에 각각 한 개씩 설치를 한다. 그럼 한번 자동차가 교차로에 진입시에, 신호등이 파란불인 동안은 계속 한 방향이 점유를 하고, 다음 신호등에 다른 방향이 점유를 하게 된다. 즉, 서로 자원을 가진 채로 다른 자원을 대기하지 않으므로, Hold and wait 한 상황이 일어나지 않는다. 또한, 가로측 방향이 신호등 때문에, 세로측 방향을 기다릴 수 있지만, 그 사이 세로측 방향은 무조건 움직 일 수 있으므로, Circular wait 한 상황이 일어나지 않는다. 그렇게 2)과 3)을 해결하여 deadlock 상황을 회피 할 수 있다.

7.22 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 unsafe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe.

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

	Allocation	Max	Need
	ABCD	ABCD	ABCD
P0	3014	5117	2103
P1	2210	3211	1001
P2	3121	3321	0200
P3	0510	4612	4102
P4	4212	6325	2113

Available 이 (0, 3, 0, 1)이므로 먼저 P2 를 해결 할 수 있다.
그럼 Available 이 (3, 4, 2, 2)가 되고 P1 을 해결 할 수 있다.
그럼 Available 이 (5, 6, 3, 2)가 되고 P3 을 해결 할 수 있다.
그럼 Available 이 (5, 11, 4, 2)가 되었지만, P0 와 P4 둘 다 D 를 3 개씩 필요로 하지만, Available 한 D 는 2 밖에 되지 않아 더 이상 진행을 할 수 없어 unsafe 한 상황이다.

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

Available 이 (1, 0, 0, 2)이므로 P1 을 해결 할 수 있다.
그럼 Available 이 (3, 2, 1, 2)이 되어 P2 를 해결 할 수 있다.
그럼 Available 이 (6, 3, 3, 3)이 되어 P0, P3, P4 모두 해결 가능해 safe 한 상태이다.
먼저 P0 을 해결하면 Available 이 (9, 3, 4, 7)이 된다.
다음 P3 을 해결하면 Available 이 (9, 8, 5, 7)이 된다.
다음 P4 를 해결하면 Available 이 (13, 10, 6, 9)가 되고 끝이 난다.
Possible solution : P1 -> P2 -> P0 -> P3 -> P4

```
❯ gcc Banker.c -o Banker.out
❯ ./Banker.out
# of Resources : 4
# of Processes : 5
Total : 12 13 6 8
Resource : 3 0 1 4 2 2 1 0 3 1 2 1 0 5 1 0 4 2 1 2
Max : 4 1 1 7 3 2 1 1 3 3 2 1 4 6 1 2 6 3 2 5
Allocated resources: 12 10 6 7
Available resources: 0 3 0 1
Process2 is on.
Available is [3 4 2 2]
Process1 is on.
Available is [5 6 3 2]
Process3 is on.
Available is [5 11 4 2]
Process is unsafe.
```

```
❯ ./Banker.out
# of Resources : 4
# of Processes : 5
Total : 13 10 6 9
Resource : 3 0 1 4 2 2 1 0 3 1 2 1 0 5 1 0 4 2 1 2
Max : 4 1 1 7 3 2 1 1 3 3 2 1 4 6 1 2 6 3 2 5
Allocated resources: 12 10 6 7
Available resources: 1 0 0 2
Process1 is on.
Available is [3 2 1 2]
Process2 is on.
Available is [6 3 3 3]
Process0 is on.
Available is [9 3 4 7]
Process3 is on.
Available is [9 8 5 7]
Process4 is on.
Available is [13 10 6 9]
Process is safe.
```