

교착 및 기아 상태

- 교착 상태

컴퓨터소프트웨어학과

김병국 교수



- 프로세스의 교착상태에 대한 개념을 안다.
- 자원할당그래프를 분석할 수 있다.
- 교착상태가 발생하는 조건을 안다.
- 교착상태를 분석할 수 있다.



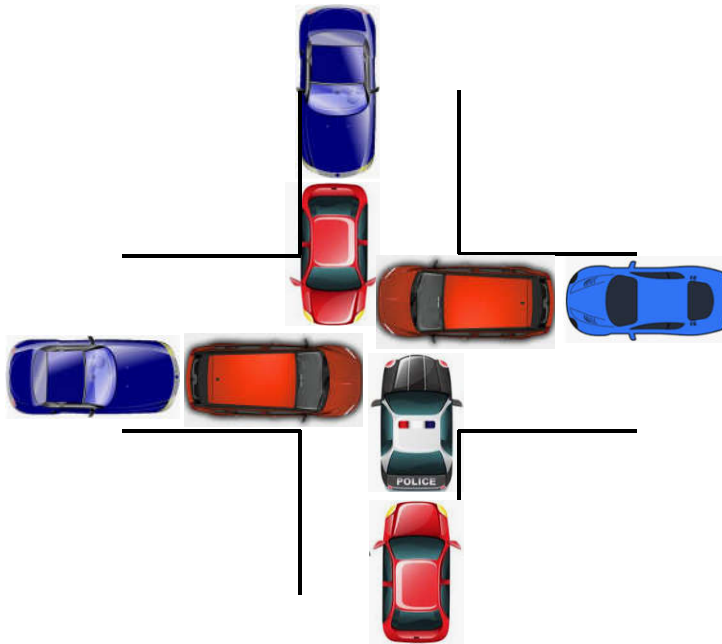
- 교착 상태
- 임계 구역에 따른 교착 상태
- 자원 할당 그래프
- 식사하는 철학자 문제
- 교착 상태 필요조건
- 교착상태 분석



1. 교착 상태 (1/4)

Dead Lock

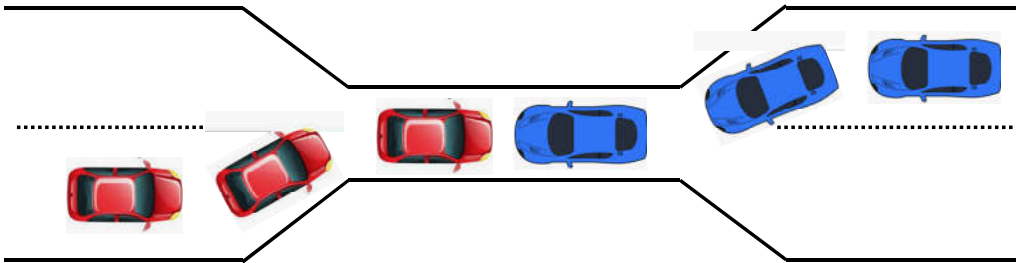
- 2개 이상의 프로세스가 서로 다른 프로세스의 작업이 끝나기만 기다리며 작업을 더 이상 진행하지 못하는 상태
- 병렬처리 기술과 자원 공유에 따라 발생된 부작용 중의 하나
 - 여러 프로세스가 작업을 진행하다 보니 자연 발생적으로 일어나는 문제
- 아사(기아) 현상 : 특정 프로세스의 작업이 끊임없이 지연되는 문제



1. 교착 상태(2/4)

□ 교착 공유 예

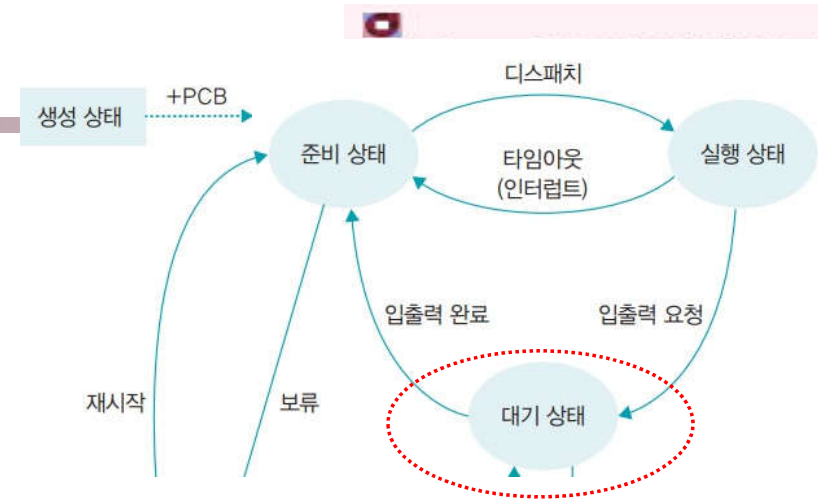
- 교각을 공유 자원(Resource)에 비유
- 교착상태가 발생하면 한쪽의 차가 후진을 해줘야 해결되는 상황
- 기아상태에 빠질 수 있음
- 대부분의 운영체제에서 교착상태를 완전히 예방해 주지는 못함



1. 교착 상태 (3/4)

□ 프로세스의 운영체제 자원의 이용 방식

- 운영체제는 프로세스에게 공유될 모든 자원들을 관리
 - 프로세스의 자원 할당을 관리
 - 각 자원에 대한 점유 상태 및 가용 상태 관리
- 모든 자원은 운영체제가 제공하는 시스템함수의 호출로 접근
 - 요청(Get / Open)
 - 자원의 접근(할당)을 요청
 - 이미 사용 중이면 대기
 - 사용(Hold / Read or Write)
 - 할당된 자원을 사용
 - 해제(Release / Close)
 - 할당된 자원을 운영체제에게 반환(return)



1. 교착 상태 (4/4)

□ 발생 원인

- 다른 프로세스와 공동으로 사용(공유)할 수 없는 시스템 자원을 사용할 때 주로 발생

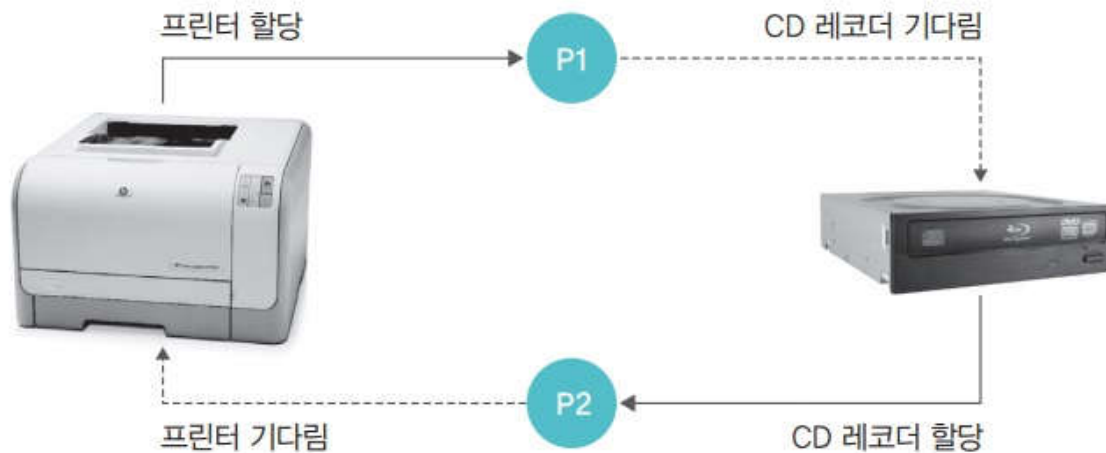


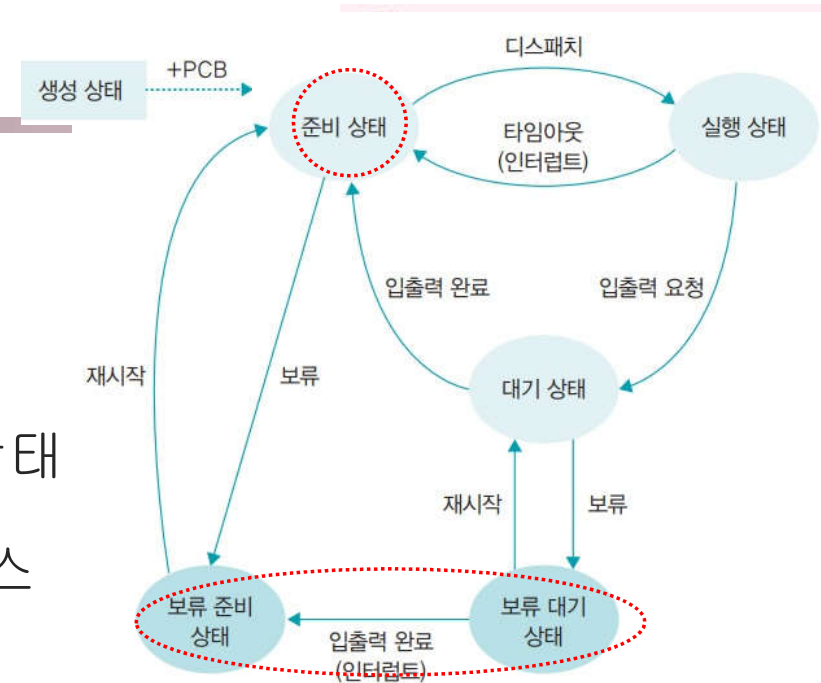
그림 6-3 시스템 자원 사용 도중 교착 상태 발생



2. 기아 상태

□ Starvation

- 무한히 대기하고 있는 상태
- 교착 상태(Dead Lock)로 인한 무한한 기다림 상태
- 스케줄링 과정에서 더 높은 우선순위의 프로세스로 인해 계속 되는 양보
- 운영체제 내 교착 상태를 예방하는 과정에서 무한한 기다림이 발생



3. 임계 구역에 따른 교착 상태 (1/2)

□ 임계 변수 값의 상호 변경 상태를 대기

- 공유 변수를 사용할 때 발생
- 데이터베이스의 경우 잠금 기능에 따른 교착상태 발생

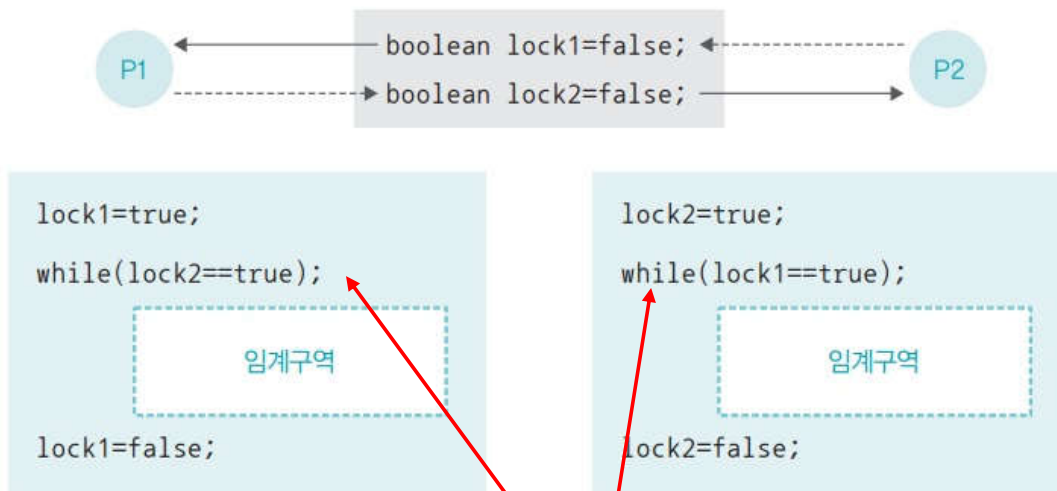


그림 6-4 교착 상태를 발생시키는 임계구역 코드

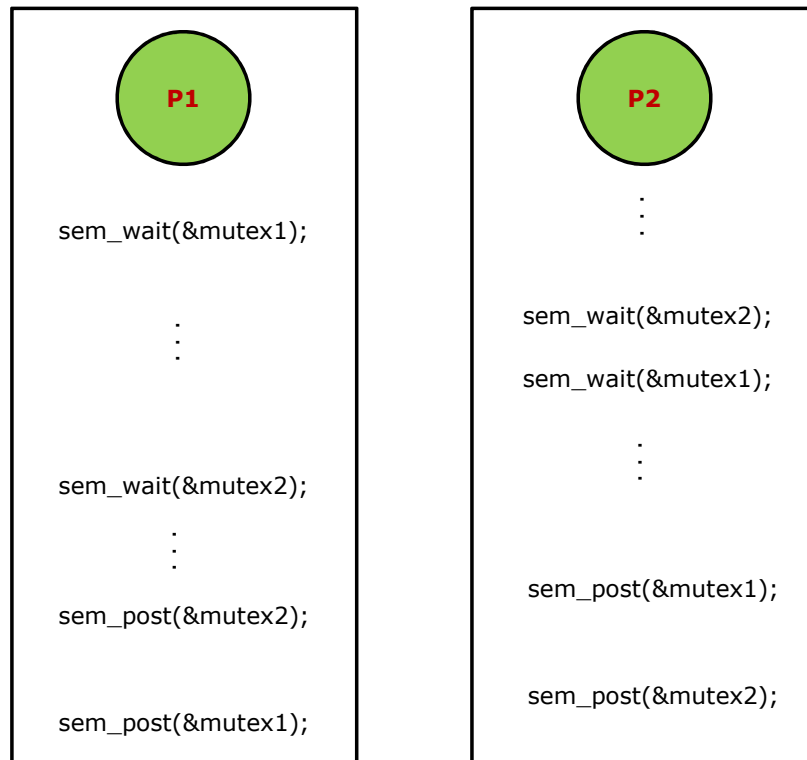
무한 반복



3. 임계 구역에 따른 교착 상태 (2/2)

□ 세마포어의 교착상태



- 서로 다른 두 세마포어 간 서로 대기상태 진입



4. 자원 할당 그래프 (1/3)

□ 자원 할당 그래프

- RAG : Resource Allocation Graph
- 프로세스가 어떤 자원을 사용 중이고 기다리는지를 표현
- 프로세스와 자원에 대하여 방향성을 갖는 화살표로 연결

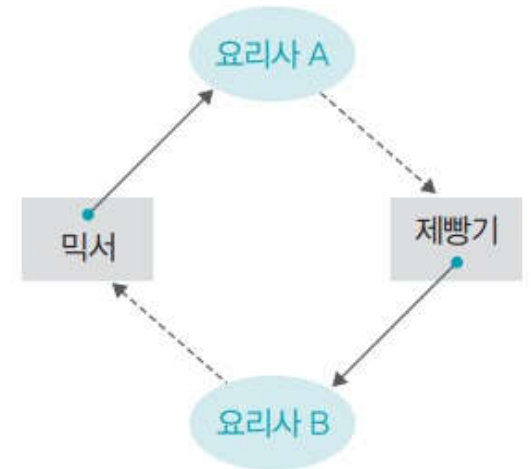
- 프로세스 = 원 
- 자원 = 사각형  ○○○...
- 기다림 : 점선 화살표→
- 할당 : 실선 화살표 —————→



프로세스 P1이 자원 R1을 할당받음



프로세스 P1이 자원 R1을 기다림



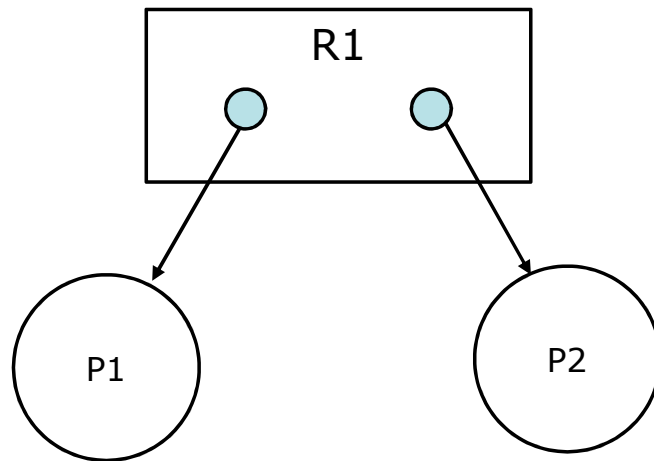
【자원 할당 그래프 예】



4. 자원 할당 그래프 (2/3)

□ 다중 자원

- 여러 프로세스가 하나의 자원을 동시에 사용하는 경우
- 수용할 수 있는 프로세스 수를 사각형 안에 작은 동그라미로 표현



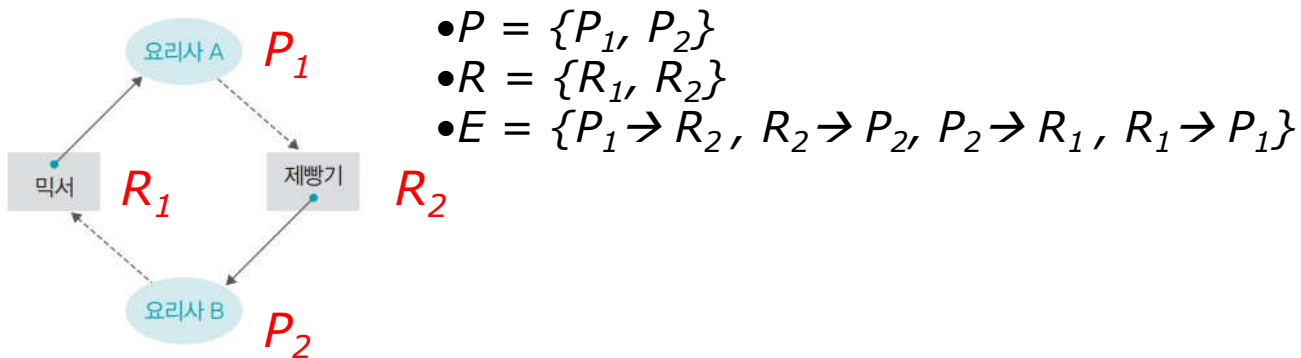
[2개의 프로세스를 수용할 수 있는 자원]



4. 자원 할당 그래프 (3/3)

□ 교착상태를 수학적으로 기술하는 방법

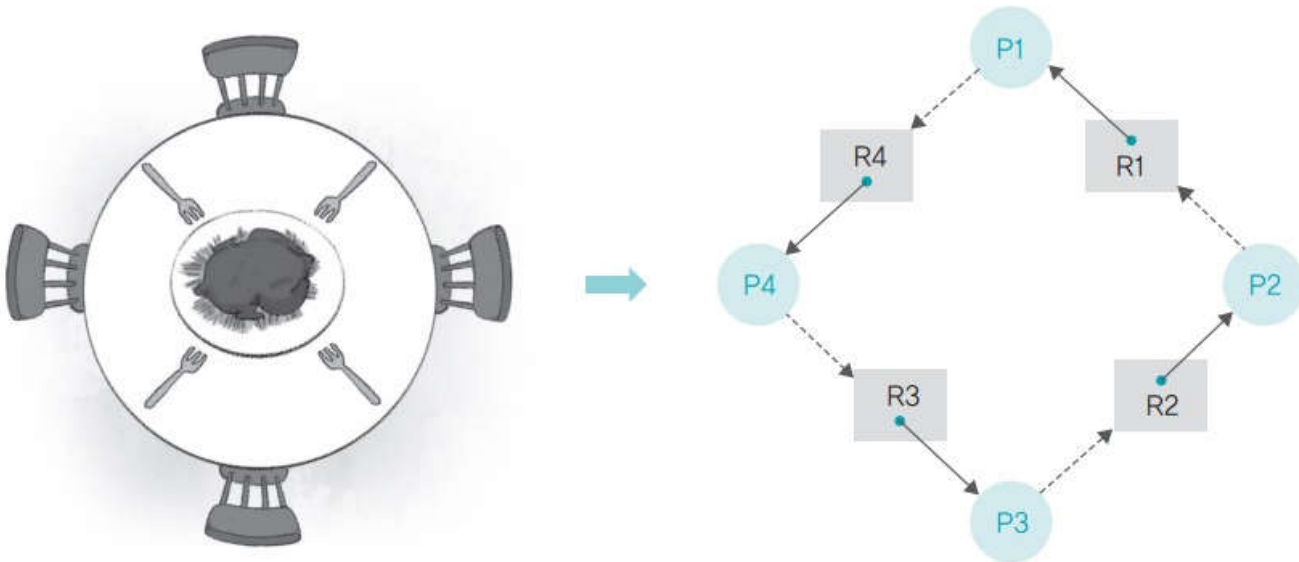
- 그래프를 수식으로 표현
- $G = (V, E)$
- V : Vertex(정점)는 Process 집합(P)과 Resource 집합(R)으로 구성
 - $P = \{P_1, P_2, P_3, \dots\}$
 - $R = \{R_1, R_2, R_3, \dots\}$
- E : Edge(간선)로 구성
 - 요청(Request Edge) : $P_i \rightarrow R_j$
 - 할당(Allocation Edge) : $R_j \rightarrow P_i$



5. 식사하는 철학자 문제 (1/3)

□ 가정

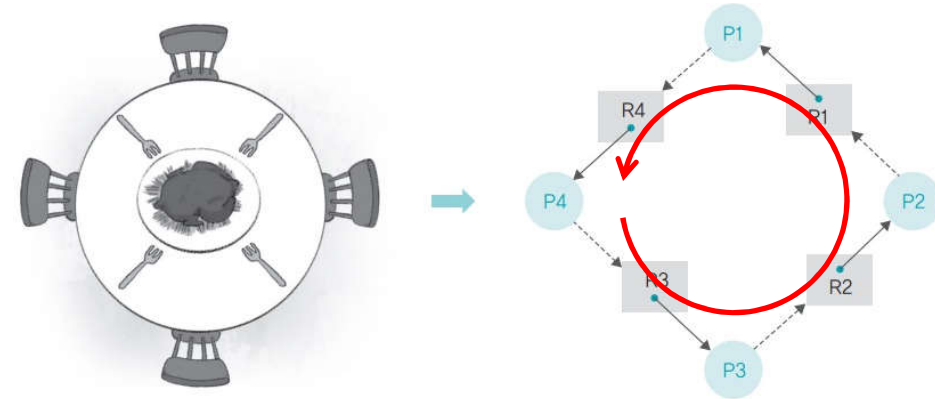
- 식사를 위해서는 두 개의 포크를 사용해야 하나, 한 순간에는 하나의 포크 만을 쓸 수 있는 상황일 때
- 왼쪽에 있는 포크를 잡은 뒤 오른쪽에 있는 포크를 잡아야만 식사 가능



5. 식사하는 철학자 문제 (2/3)

□ 발생 조건

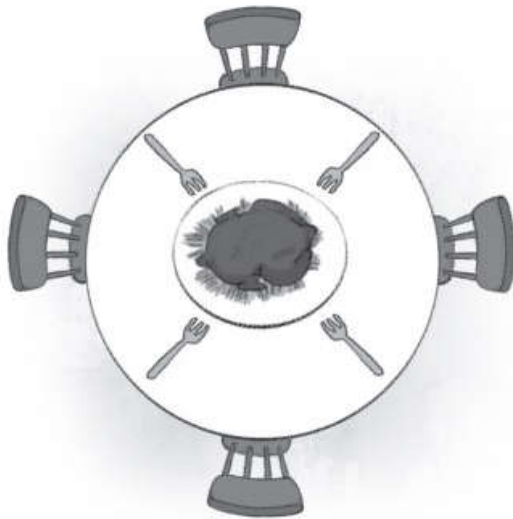
- 철학자들은 서로 포크를 공유할 수 없음
 - 자원을 공유하지 못하면 교착 상태 발생
- 각 철학자는 다른 철학자의 포크를 빼앗을 수 없음
 - 누군가 자원을 놓을 때까지 기다려야 하므로 교착 상태가 발생
- 각 철학자는 왼쪽 포크를 잡은 채 오른쪽 포크를 기다림
 - 자원 하나를 잡은 상태에서 다른 자원을 기다리면 교착 상태가 발생
- 자원 할당 그래프가 원형(Circle)
 - 자원을 요구하는 방향이 원을 이루면 양보를 하지 않기 때문에 교착 상태가 발생



5. 식사하는 철학자 문제 (3/3)

□ 문제 해결책

- 포크 개수보다 철학자 수를 줄임
- 양쪽에 포크가 사용 가능할 때, 두 포크를 점유
- 홀수 번 위치와 짝수 번 위치 간의 포크의 잡는 순서를 달리함



6. 교착 상태 필요조건

□ 필요조건

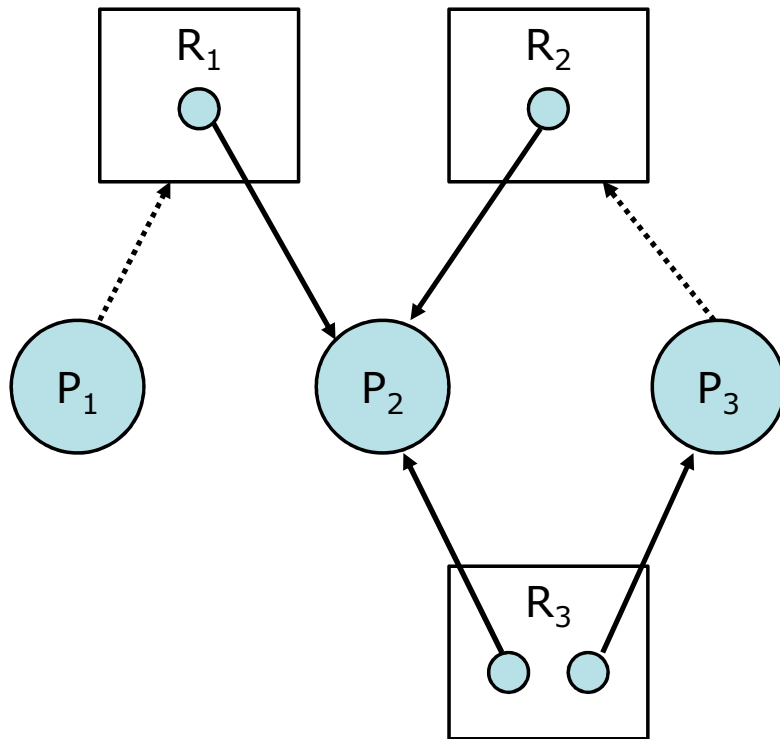
- 상호 배제(mutual exclusion)
 - 한 프로세스가 사용하는 자원은 다른 프로세스와 공유할 수 없는 배타적인 자원
- 비선점(non-preemptive)
 - 사용 중인 자원은 중간에 다른 프로세스가 빼앗을 수 없어야 함
- 점유와 대기(hold and wait)
 - 프로세스가 어떤 자원을 할당 받은 상태에서 다른 자원을 기다리는 상태
- 원형 대기(circular wait)
 - 점유와 대기를 하는 프로세스 간의 관계가 **원**을 이루어야 함



7. 교차상태 분석 (1/3)

□ 그래프 예 (1/3)

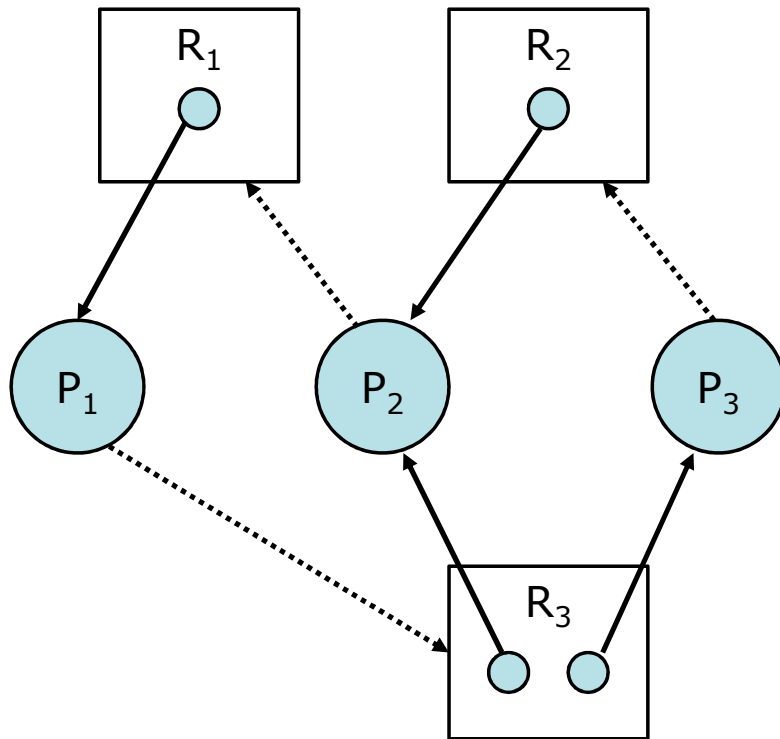
- 교차상태 발생?



7. 교차상태 분석 (2/3)

□ 그래프 예 (2/3)

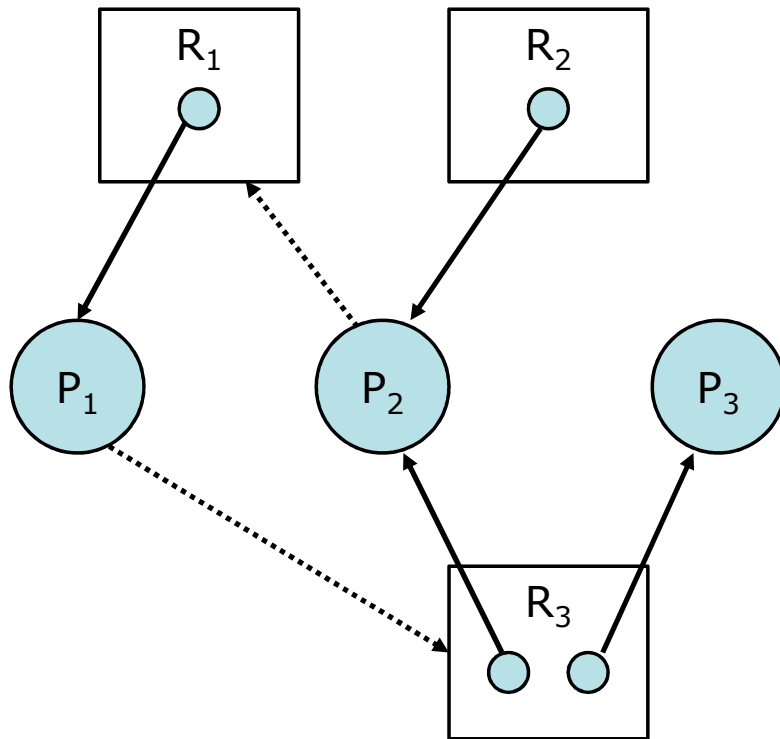
- 교차상태 발생?



7. 교차상태 분석 (3/3)

□ 그래프 예 (3/3)

- 교차상태 발생?



수고하셨습니다.

