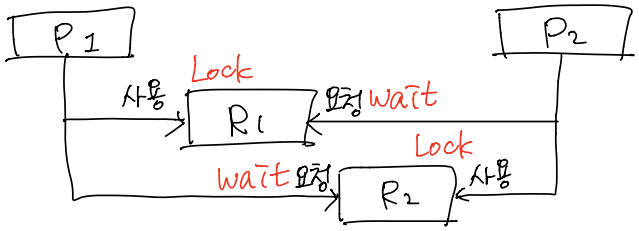
1. 데드락 (교착 상태) 개념

- 한정된 자원을 여러 프로세스가 동시에 사용하고자할 때, 서로 원하는 자원을 얻지 못하고 계속 상대방의 작업이 끝나기만을 기다리는 상황.



⇒ 서로 요청하는 자원을 점유해야만 락을 걸고 있는 자원을 릴리즈하려 함.
(무한정 대기해야하는 상황)

2. 데드락 발생 조건. (4가지 조건 모두 성립해야)

① 상호 배제 (Mutual Exclusion) : 한 번에 하나의 프로세스
② 점유 대기 (Hold and wait) : 하나 가진채 다른 자원 요청.
③ 비선점 (No Preemption) : 빼앗을 수 없다.
④ 순환 대기 (Circular wait) : $\{P_0 \sim P_{n-1}\}$ 에서 $P_{n-1} \rightarrow P_n$의 자원을 대기, P_n은 P_0의 자원을 대기.

3. 데드락 대처 방법

① 예방 및 회피 : 발생하지 않도록 예방, 회피.
② 탐지 및 회복 : 발생하면 탐지, 해결.
③ 무시.

- 예방 : 주로 "④순환 대기 부정" 방법 사용하여 예방

① 상호 배제 부정 : 여러 프로세스가 자원 사용 허용.
② 점유 대기 부정 : 프로세스 실행 전 모든 필요한 자원 할당한다. ← 이 두개 헷갈림.
③ 비선점 부정 : 다른자원 요구 시, 점유 중인 자원 반납하고 요구. (대기)
④ 순환 대기 부정 : 자원에 고유한 번호를 할당 → 번호순서대로 자원을 요구.

- 회피 : 발생할 것 같으면 회피하도록.

⇒*은행원 알고리즘 (Banker's Algorithm)

: 자원 요청 → 안정여부 체크 → 안정하면 할당, 아니면 거부.

↳ 프로세스가 자원 요구 시 할당 후에도 안정 상태인지 검사.

⇒은행원 알고리즘 단점

① 할당할 수 있는 자원의 수가 일정해야함.
② 사용자 수가 일정해야함
} 자원&사용자 수에 영향 (제한적).

*③ 항상 불안정 상태 방지해야하므로 자원 이용도 ↓

*은행원 알고리즘에서 OS는 안정 상태 유지 가능할 때만 수락 X면 요구 계속 거절

⇒ Max, Allocated, Need, Available을 통해 요청들에 대해 할당해주고 반환받았을 때, 최소한 다른 한 프로세스에 할당할 자원이 남는지 검사하는 것.

-탐지: *자원할당 그래프를 통해 데드락을 탐지 후 해결하려는 것.

⇒ 단, 자원 요청 시마다 '데드락 탐지' 시 오버헤드 발생.

* P와 R의 방향 그래프를 통해 '순환 대기' 상태 파악

-회복 : 데드락 상태의 프로세스를 종료하거나, 할당된 자원을 해제.

1) 종료 : 데드락 일으킨 P를 하나씩 종료 (데드락 X일 때 까지)

2) 선점 : OS가 데드락에 빠진 자원을 선점해 다른 P에 할당.
(해당 P는 일시정지)

-무시: 데드락은 자주 발생 X, 예방·회피, 탐지·회복은 비용↑.

(UNIX, MINIX에서 이 방법 사용).