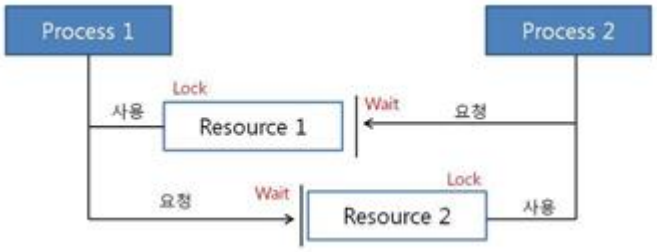
< 데드락 (DeadLock, 교착 상태) >

두 개 이상의 프로세스나 스레드가 서로 자원을 얻지 못해서 다음 처리를 하지 못하는 상태, 무한히 다음 자원을 기다리게 되는 상태를 말한다.

시스템적으로 한정된 자원을 여러 곳에서 사용하려고 할 때 발생한다.

(마치, 외나무 다리의 양 끝에서 서로가 비켜주기를 기다리고만 있는 것과 같다.)

1. 데드락이 일어나는 경우



프로세스1과 2가 자원1, 2를 모두 얻어야 한다고 가정해보자

t1 : 프로세스1이 자원1을 얻음 / 프로세스2가 자원2를 얻음

t2 : 프로세스1은 자원2를 기다림 / 프로세스2는 자원1을 기다림

현재 서로 원하는 자원이 상대방에 할당되어 있어서 두 프로세스는 무한정 wait 상태에 빠짐 (Dead Rock)

(주로 발생하는 경우)

- 멀티 프로그래밍 환경에서 한정된 자원을 얻기 위해 서로 경쟁하는 상황 발생
- 한 프로세스가 자원을 요청했을 때, 동시에 그 자원을 사용할 수 없는 상황이 발생할 수 있음. 이때 프로세스는 대기 상태로 들어감
- 대기 상태로 들어간 프로세스들이 실행 상태로 변경될 수 없을 때 '교착 상태' 발생

2. 데드락(DeadLock) 발생 조건

4가지 모두 성립해야 데드락 발생

(하나라도 성립하지 않으면 데드락 문제 해결 가능)

1) 상호 배제(Mutual exclusion)

자원은 한 번에 한 프로세스만 사용할 수 있음

2) 점유 대기(Hold and wait)

최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용하고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 존재해야 함

3) 비선점(No preemption)

다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없음

4) 순환 대기(Circular wait)

프로세스의 집합에서 순환 형태로 자원을 대기하고 있어야 함

3. 데드락(DeadLock) 처리

1) 교착 상태를 예방 & 회피

- **예방(prevention)**

  교착 상태 발생 조건 중 하나를 제거하면서 해결한다 (자원 낭비 엄청 심함)

- 상호배제 부정 : 여러 프로세스가 공유 자원 사용
- 점유대기 부정 : 프로세스 실행전 모든 자원을 할당
- 비선점 부정 : 자원 점유 중인 프로세스가 다른 자원을 요구할 때 가진 자원 반납
- 순환대기 부정 : 자원에 고유번호 할당 후 순서대로 자원 요구

- **회피(avoidance)**

교착 상태 발생 시 피해 나가는 방법

은행원 알고리즘(Banker's Algorithm)

- 은행에서 모든 고객의 요구가 충족되도록 현금을 할당하는데서 유래함
- 프로세스가 자원을 요구할 때, 시스템은 자원을 할당한 후에도 안정 상태로 남아 있게 되는지 사전에 검사하여 교착 상태 회피
- 안정 상태면 자원 할당, 아니면 다른 프로세스들이 자원 해지까지 대기

4. 교착 상태를 탐지 & 회복

교착 상태가 되도록 허용한 다음 회복시키는 방법

- **탐지(Detection)**

자원 할당 그래프를 통해 교착 상태를 탐지함

자원 요청 시, 탐지 알고리즘을 실행시켜 그에 대한 오버헤드 발생함

- **회복(Recovery)**

교착 상태 일으킨 프로세스를 종료하거나, 할당된 자원을 해제시켜 회복시키는 방법

프로세스 종료 방법

- 교착 상태의 프로세스를 모두 중지
- 교착 상태가 제거될 때까지 하나씩 프로세스 중지

자원 선점 방법

- 교착 상태의 프로세스가 점유하고 있는 자원을 선점해 다른 프로세스에게 할당 (해당 프로세스 일시정지 시킴)
- 우선 순위가 낮은 프로세스나 수행 횟수 적은 프로세스 위주로 프로세스 자원 선점

Reference

https://github.com/gyoogle/tech-interview-for-developer/blob/master/Computer%20Science/Operating%20System/DeadLock.md