Effective Java 읽기 스터디 10min Mini Seminar

7주차 2021.04.12

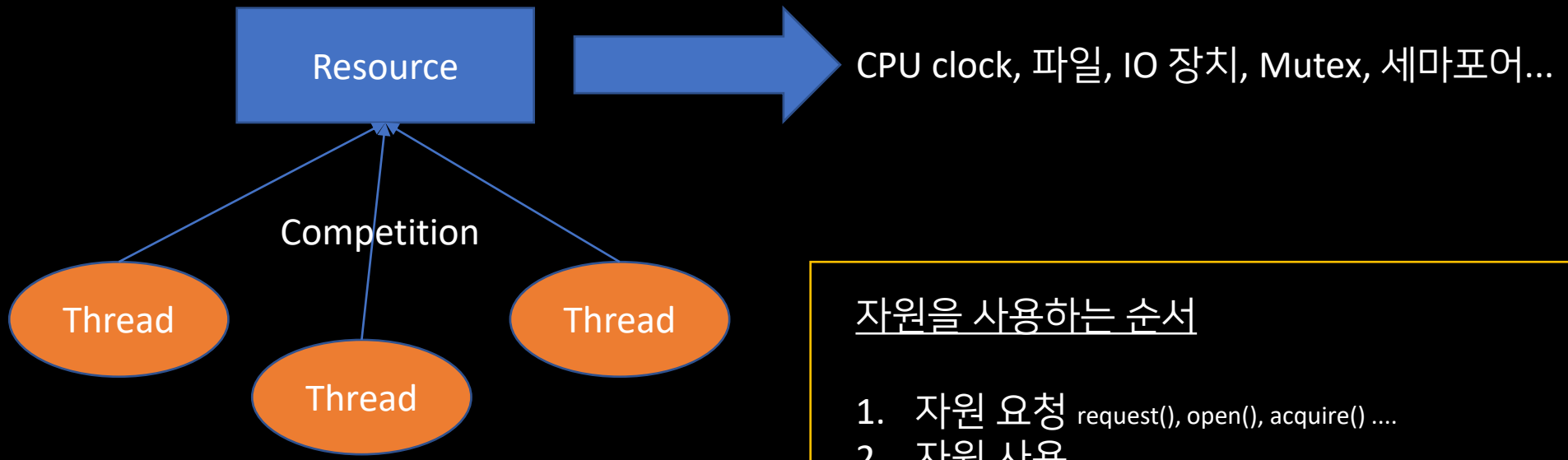
Deadlock

발표자: 김정수

Contents

1. 교착 상태
2. Deadlock의 조건
3. Deadlock 처리 방법

1. 교착 상태



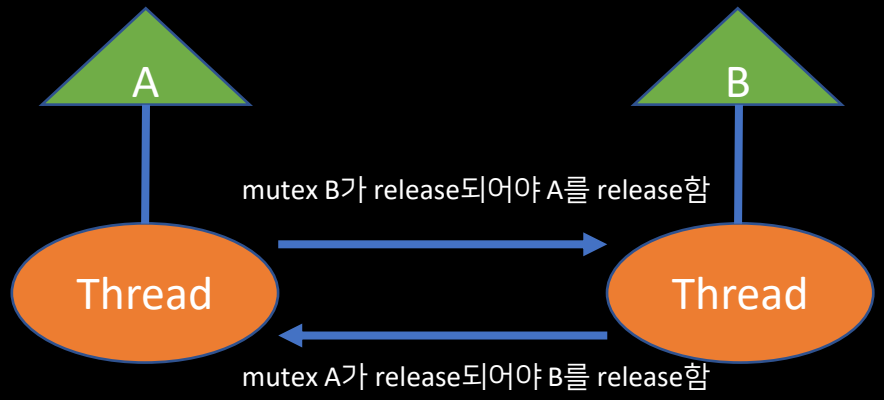
자원을 사용하는 순서

1. 자원 요청 request(), open(), acquire() ....
2. 자원 사용
3. 자원 방출 release(), close(), free() ...

자원을 사용하는 중
다른 스레드가 자원에 접근할 경우
동기화 문제 발생

→ 대부분의 경우
한번에 한 스레드만 자원에 접근 가능하도록 함.

1. 교착 상태



mutex A 획득
mutex B 획득
작업 수행
mutex B 반납
mutex A 반납

Race Condition

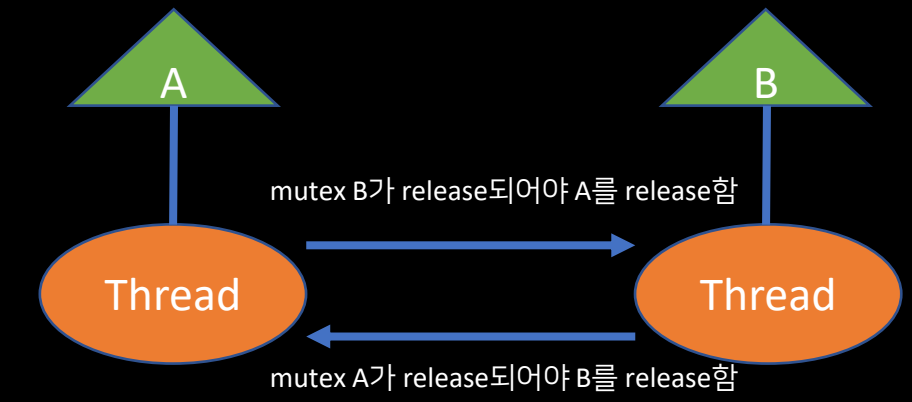
mutex B 획득
mutex A 획득
작업 수행
mutex A 반납
mutex B 반납

Deadlock:

한 스레드 집합 안의 모든 스레드가
집한 안의 다른 스레드에 의해서만
발생될 수 있는 이벤트를 무한정 기다리는 상태

ex. mutex lock을 release하는 이벤트

1. 교착 상태



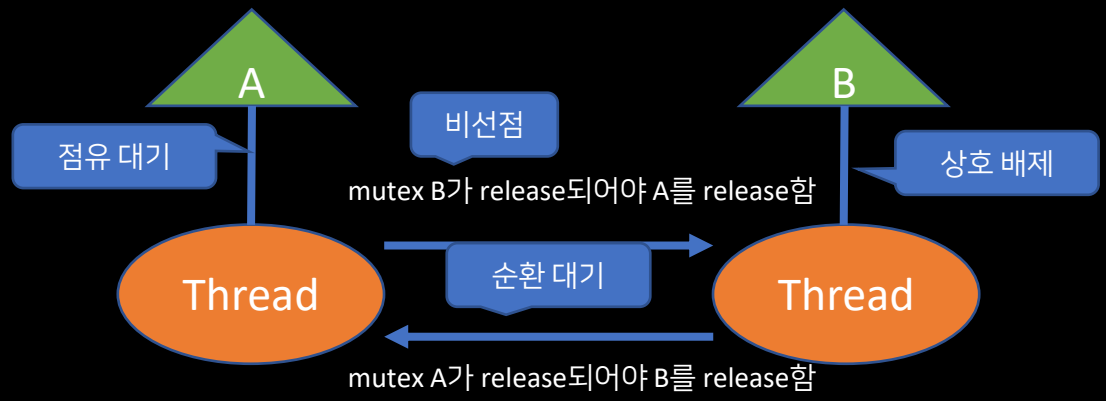
Livelock:

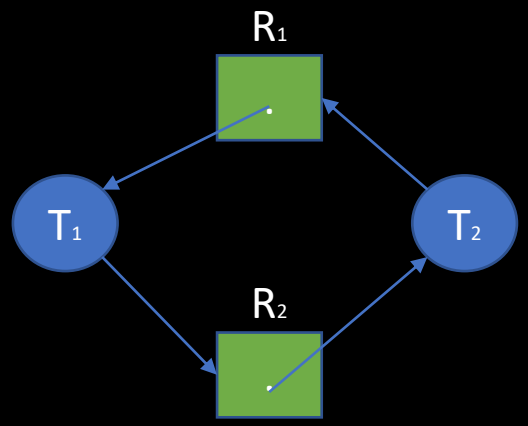
스레드가 실패한 행동을 계속해서 시도할 때 발생한다.

Race Condition발생 시 자신의 lock을 release하지만,
상대도 똑같이 release하기 때문에 같은 행동을 반복한다.

2. Deadlock 조건

1. 상호 배제 Mutual Exclusion
2. 점유 대기 Hold-and-Wait
3. 비선점 No preemption
4. 순환 대기 Circular wait





cycle 발생!!

3. Deadlock 처리 방법

1. 무시(ignorance)
2. 예방(prevention)
3. 회피(avoidance)
4. 검출(detection), 복구(recover)

무시(ignorance)

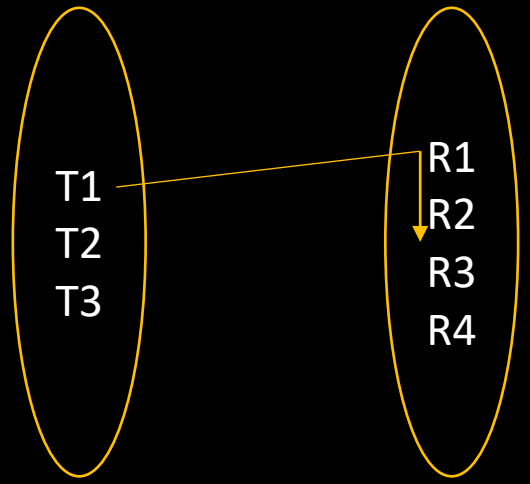
교착 상태가 발생해도 무시한다.
시스템을 정지시키고 수작업으로 다시 시작한다.

⇒ 비용이 적게 든다.

ex) Linux, Windows 등 OS

3. Deadlock 처리 방법

1. 무시(ignorance)
2. 예방(prevention)
3. 회피(avoidance)
4. 검출(detection), 복구(recover)



R1, R2를 사용하고 싶으면
R1를 먼저 요청한 뒤 R2를 요청해야 한다.

예방(prevention)

Deadlock 필요 조건 4가지 중 최소 하나를 제거한다.

- Mutual Exclusion 제거
  a) read-only로 만든다.
- Hold and Wait 제거 (starvation 발생 가능)
  a) 실행 시작 전에 모든 자원을 요청하고 할당한다.
  b) 자원을 갖고 있지 않을 때만 요청하도록 한다.
- No Preemption 제거
  a)스레드가 대기해야 할 경우 모든 자원을 사용할 수 있을 때 다시 시작한다. (대기 시 점유 자원을 선점-방출)
  => mutex lock이나 세마포어에 적용 불가
- Circular wait 제거
  a) 순서대로 자원을 요청하도록 함.

=> 장치의 이용률이 저하되고 throughput이 감소한다.

3. Deadlock 처리 방법

1. 무시(ignorance)
2. 예방(prevention)
3. 회피(avoidance)
4. 검출(detection), 복구(recover)

회피(avoidance)

자원이 어떻게 요청될지 파악해서 Deadlock을 발생시키지 않도록 한다.

a) 자원 할당 그래프 알고리즘
: 사이클 탐지 알고리즘을 사용해 사이클을 형성하지 않을 때만 요청을 허용한다.
=> 종류마다 자원이 여러 개 있으면 사용할 수 없다.

b) 은행원 알고리즘
: 스레드가 자원을 요청할 때 요청을 수락할 경우에도 시스템이 안전 상태를 유지하는지 판단한다.
=> 종류마다 자원이 여러 개 있어도 사용할 수 있다.

3. Deadlock 처리 방법

1. 무시(ignorance)
2. 예방(prevention)
3. 회피(avoidance)
4. 검출(detection), 복구(recover)

검출(detection), 복구(recover)

교착 상태 예방/방지 알고리즘이 없을 경우 반드시 필요하다.

- 교착상태 검출/탐지

wait-for graph(DB) or 은행원 알고리즘과 유사한 알고리즘 사용.

- 복구

a) 교착 상태 프로세스를 모두 중지시킨다.
b) 교착 상태가 제거될 때까지 하나씩 중지시킨다.
c) 자원 선점을 이용해 Deadlock이 제거될 때까지
자원을 계속 선점해 다른 프로세스에게 준다.
-> 희생자 선택, 후퇴(rollback), 기아 상태 고려해야 함.

References

1~3)
Operating System Concepts 10/E

Q&A