병행제어 2

데드락

여러개의 프로세스 혹은 스레드가 서로 가지고있는 자원을 무한히 기다리는 현상

A, B, C, D

자원 반납 대상 - 기다리는 대상
A - B
B - C
C - D
D - A

해결책

- 자원 얻는 순서를 정해두면 문제 해결 가능

Starvation (기아)

무한정 기다림

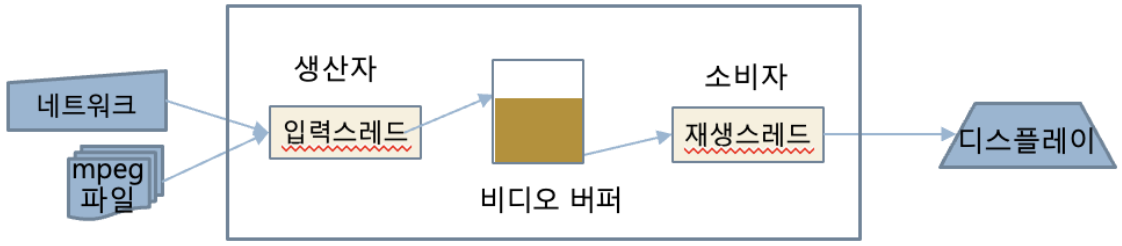
세마포에서 일어날 수 있는 문제점

1. Bounded-Buffer Problem (생산자 소비자 문제)
2. Readers and Writers Problem
3. Dining-Philosopheres Problem (데드락)

생산자 소비자 문제

멀티스레드 응용 프로그램에서 발생할 수 있는 문제

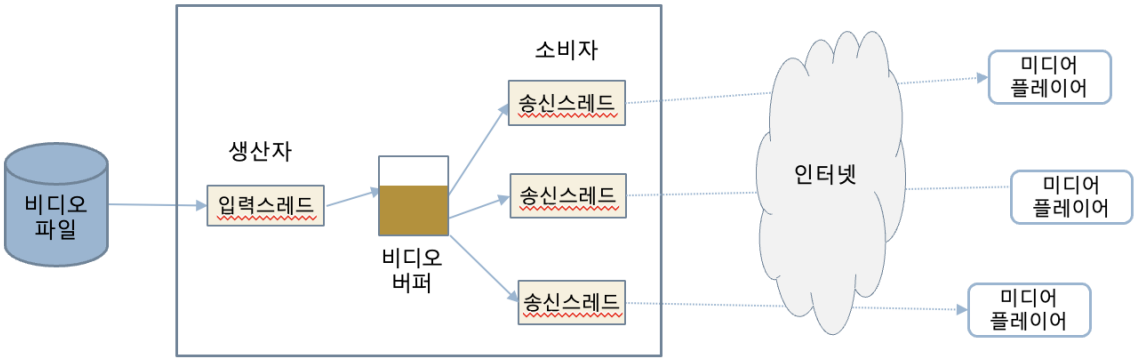
사례 1.



비디오 플레이어 응용프로그램

- 입력 스레드가 네트워크나 동영상 파일로부터 한 프레임 씩 읽어서 비디오 버퍼에 저장하는 일을 반복
- 재생 스레드는 비디오 버퍼에 도착하는 프레임을 읽고 디코딩하여 디스플레이에 출력

사례 2.



스트리밍 서버 응용프로그램

- 입력스레드는 비디오 파일로부터 비디오 프레임을 읽어서 비디오 버퍼에 공급
- 송신스레드는 비디오 버퍼로부터 프레임을 읽어서 미디어 플레이어로 전송하는 작업

입력스레드 : 공유버퍼에 데이터를 공급하는 생산자

재생, 송신스레드 : 공유버퍼에서 데이터를 읽고 소비하는 생산자

발생하는 문제

1. 상호배제의 문제(생산자들과 소비자들의 동시적 공유버퍼 사용에 대한 상호배제)
2. 비어있는 공유버퍼 문제 - 소비자가 비어있는 공유버퍼를 읽는 경우
3. 가득찬 공유버퍼에 생산자가 데이터를 공급하는 경우

상호배제 문제 해결

- 생산자와 소비자가 하나씩 있다면 둘 사이에서 공유버퍼를 접근하는 임계구역에서 문제 발생
- 여러개가 존재한다면 생산자간 혹은 소비자간에도 발생할 수 있다.
- 임계구역에 대한 상호배제를 뮤텍스 혹은 세마포를 통해 해결

비어있는 공유버퍼 문제 해결

- 버퍼가 비어있는지 확인하는 세마포의 P/V연산으로 해결 가능
- 소비자상황에서 버퍼에서 데이터를 읽기전 P연산 실행으로 버퍼가 비어있는 경우 소비자는 대기하도록

꽉찬 공유버퍼에 생산자가 데이터를 공급하는 경우

- 버퍼가 가득차 있는지 확인하는 P/V연산으로 해결가능
- 생산자상황에서 버퍼에 쓰기전에 P연산 실행해서 버퍼가 가득찬 상태면 생산자는 대기하도록

W : 카운팅 세마포. 버퍼에 있는 쓰기 가능한 저장 공간의 개수가 0이면(꽉차있는 경우) wait
R : 카운팅 세마포. 버퍼에 읽기 가능한 저장 공간의 개수가 0이면(비어있는 경우) wait
M : 뮤텍스로서 생산자 소비자의 모든 스레드에 의해 사용

```
Consumer { // 소비자 스레드
   while(true) {
      P(R); // 세마포 R에 P/wait 연산을 수행하여
            // 버퍼가 비어 있으면(읽기 가능한 버퍼 수=0) 대기한다.

      뮤텍스(M)를 잠근다.
      공유버퍼에서 데이터를 읽는다. // 임계구역 코드
      뮤텍스(M)를 연다.

      V(W); // 세마포 W에 대해 V/signal 연산을 수행하여
            // 버퍼가 비기를 기다리는 Producer를 깨운다.
   }
}
```

```
Producer { // 생산자 스레드
   while(true) {
      P(W); // 세마포 W에 P/wait 연산을 수행하여
            // 버퍼가 꽉 차 있으면(쓰기 가능한 버퍼 수=0) 대기

      뮤텍스(M)를 잠근다.
      공유버퍼에 데이터를 저장한다. // 임계구역 코드
      뮤텍스(M)를 연다.

      V(R); // 세마포 R에 대해 V/signal 연산을 수행하여
            // 버퍼에 데이터가 저장되기를 기다리는 Consumer를 깨운다.
   }
}
```

생산소비자 문제는 뮤텍스, 세마포를 활용하여 해결하는데 이것이 세마포에서 일어날 수 있는 문제인건가?

Readers and Writers Problem

임계구역에 접근하는 프로세스가 두 종류 상황

- reader
- writer

발생하는 문제

- write 상황시에만 상호배제함

- reader는 임계구역에 동시 접근이 가능하여 한 스레드가 write하는 상황에서 read하면 문제가 발생

해결 방법

1. reader에게 우선순위를 주어 reader가 임계구역에 들어가있지 않다면 writer 진입 허용
2. writer에게 우선순위를 주어 writer가 임계구역에 들어가있지 않다면 reader진입 허용
3. 우선순위 없이 작업 순서대로 진행

Dining-Philosopheres Problem (데드락)

설명 : 데드락은 교착상태로 상대방이 가진 자원을 반납할때 까지 무한히 기다리는 상황

> 💡 상황
>
> 원형테이블
> 철학자 A, B, C, D 앉음
> 포크를 4개, 음식 준비
> 음식은 한사람이 포크를 두개 가지고 있을때 먹을 수 있음
> A, B, C, D 서로 상대방의 포크만 보면서 밥을 못먹고 가만히 대기하는 상황

교착상태의 발생 조건

- 상호 배제
  - 공유 자원에 대해 락을 걸어 다른 프로세스가 접근하지 못하게 해야함
- 점유와 대기
  - 공유 자원을 점유한 상태에서 다른 공유 자원을 사용하기 위해 대기하고 있는 상황
- 비선점
  - 다른 프로세스가 점유한 자원을 강제로 빼앗을 수 없는 상황
- 환형 대기

    - 순환, 원형 형태로 서로 필요한 자원을 점유하고 대기하는 상황

⇒ 위의 4가지 상황을 하나라도 발생하지 않게 않다면 데드락은 발생하지 않는다.

우선순위 역전

문제점

- 실시간 시스템에서의 문제
- 우선순위가 높은 것을 먼저 실행시켜야 하는 것이 당연한 것인데 우선순위에 따르지 않는 상황이 발생한다는 것 자체에 문제

해결방법

- 공유자원을 소유한 스레드의 우선순위를 높은 우선순위로 일시적으로 올려서 공유자원이 빼앗기지 않고 끝까지 유지되면서 실행될 수 있도록 한다.
- 이것을 우선순위 올림, 우선순위 상속이라고 한다.

모니터

- 세마포는 코딩하기 힘들고, 정확성의 입증이 어렵고, 한번의 실수가 시스템에 많은 영향을 미침
- 공유데이터를 중심으로 함수가 정의되어있다.
- 자바에서 모니터 활용
    - synchronized : 공통 자원에 접근하는 함수임을 선언
    - wait(), notify() : 실행 순서를 제어할 수 있음
        - wait() : 대기
        - notify() : 임계구역 실행종료를 알리고 대기큐에서 기다리는 스레드를 실행