OS - 프로세스들

1. 실행되는 프로세스들의 통신

배경

파일 입출력에서부터 프로그램 통신까지 프로세스 통신은 자주 사용되고 있다.

쓰레드와는 다르게 프로세스는 메모리 영역을 공유하고 있지 않아서, 데이터를 주고 받을 수 있는 통신이 필요하다.

정의

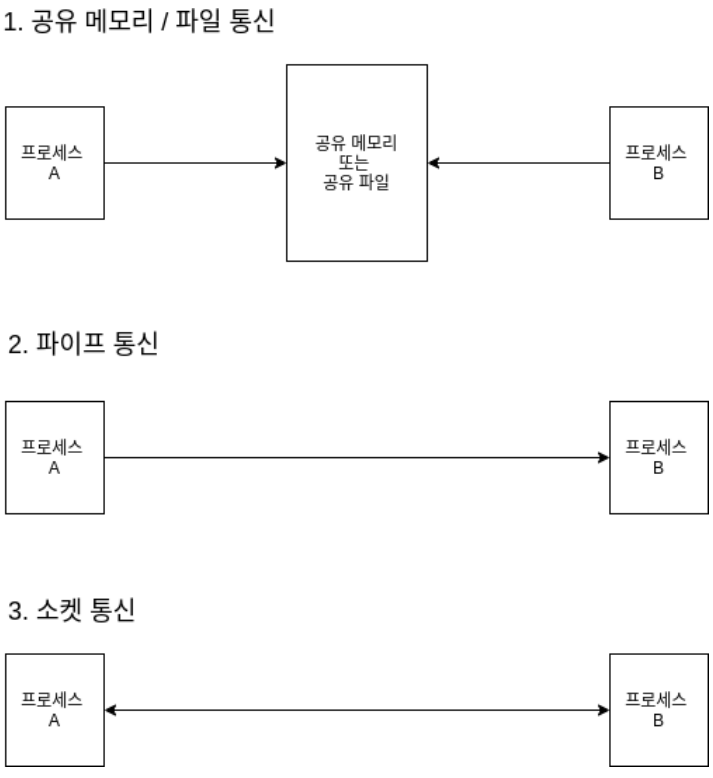
프로세스간 통신 (IPC, Inter Process Communication) 은 데이터를 주고받을 수 있는 프로세스간 통신이다. 이때, 통신이란 데이터를 주고 받는 것이다.

IPC는 거의 비슷한 흐름으로 진행되는데, open → read / write → close 순서로 진행된다.

분류

1. 비동기 통신 : 데이터 받는 쪽이 바쁜 대기 - 상태 변화를 보기 위한 무한 반복 실행으로 자원 낭비가 큰 실행 방법 - 를 사용하여, 데이터가 도착했는지 직접 확인한다.
2. 동기 통신 : 데이터 받는 쪽이 데이터 도착할 때까지 자동 대기 상태에 있는다.

종류



- 파이프란, 운영체제가 지원하는 프로세스 통신 기능으로, 동기 단방향 통신이다.
- 소켓이란, 여러 프로세스들이 통신할 수 있는 통신기법으로, 동기 양방향 통신이다.

2. 실행되는 프로세스들의 자원 동시 접근

배경

프로세스들이 자원을 공유하는 경우는 많다. 예를 들어, 같은 컴퓨터 내에서는 저장장치나 메모리를 공유할 수 있고, 다른 컴퓨터들끼리는 원격으로 MySQL, 캐시 등과 같은 자원을 공유할 수 있다.

이때, 공유되는 자원을 **공유 자원** (shared resource) 이라고 하는데, 이는 여러 프로세스가 공동으로 이용하는 변수, 메모리, 파일 등을 말한다.

문제

공유 자원으로 다음과 같은 문제들이 발생할 수 있다.

1. 경쟁 조건 (race condition) - 여러 프로세스가 공유 자원을 병행해서 읽거나 쓰는 상황 - 이 발생한다.
2. 임계 구역 (critical section) - 공유 자원의 접근 순서에 따라 실행 결과가 달라지는 프로그램 역역 - 문제가 발생한다.

이 문제는 여러 프로세스가 임계 구역을 동시 접근하지 못하도록 하여 해결할 수 있고, 이는 코드 영역에서 잠금 (LOCK) 을 활용하여 구현할 수 있다.

해결

분석1. 동시 접근하지 못하도록 할 수 있는 조건들

프로세스가 임계구역에 동시에 접근하지 못하도록 하기 위해서는, 다음과 같은 조건들을 만족시키면 된다.

(아래 코드는 각 상황을 만족시키지 못하는 코드이다.)

1. **상호 배제** : 임계 구역에는 하나의 프로세스만 들어가야 한다

```
while (lock == true)
lock = true
// 임계구역 : 하나의 프로세스가 1째줄 끝나고 타임아웃이 걸리면, 다수 프로세스들이 접근 가능하다.
lock = false
```

2. **한정 대기** : 어떤 프로세스든 임계 구역에 진입하지 못하는 무한대기 (= **데드록**) 를 하지 말아야 한다.

```
lock1 = true
while (lock2 == true)
// 임계구역 : 다수 프로세스가 1째줄 끝나고 타임아웃이 걸리면, 해당 프로세스들은 접근을 못한다.
lock1 = false
```

3. **진행의 융통성** : 한 프로세스가 다른 프로세스의 진행을 방해하면 안된다.

```
while (lock == 1)
// 임계구역 : 하나의 프로세스가 임계 구역 진행시 타임아웃이 걸리면, 다른 프로세스는 접근을 못한다.
lock = 1
```

분석2. 위의 3가지 조건들을 만족시키는 알고리즘

이를 해결하기 위한 알고리즘은 다음과 같다.

1. **피터슨 알고리즘**
   - 특징 : `turn` 변수를 선언하여, 2개의 `lock` 이 모두 걸려도, 접근이 가능하다.
   - 단점 : `lock` 변수를 프로세스 수 만큼 선언해야 한다.

```
lock1 = true
turn = 2
while (lock2 == true && turn == 2)
// 임계구역
lock1 = false
```

2. 데커 알고리즘

- 단점 : 알고리즘이 너무 복잡하다.

```
// 넘복잡
```

3. **세마포어 알고리즘**
   - 정의 : 세마포어는 공용 리소스를 잠그는 데 사용하는 보호된 변수로, 공용 자원의 개수를 값으로 갖고 있다.
   - 특징 : 공유 리소스에 접근할 수 있는 프로세스의 최대 허용치만큼만 사용자가 동시 접근할 수 있다.
   - 장점 : 바쁜 대기를 하지 않아도 된다.
   - 단점 : 연산 중간에 타임아웃 시, 다른 프로세스는 접근을 하지 못한다.

```
Semaphore(n)
P()
// 임계구역
V()


// 구현1. Semaphore(n) : n개의 공유 자원이 있다.
RS = n

// 구현2. P() : 자원 하나를 사용하고 있다.
if (RS > 0): RS = RS - 1
else : block()

// 구현3. V() : 임계구역이 비어있다.
RS = RS + 1
wake_up() // block() 을 푼다
```

분석3. 개발자 코드 외에 외부 자원들의 자체 락

다수의 프로세스들이 자원에 동시 접근할 때 개발자는 코드로 락을 제어할 수도 있지만, 자원 자체에서도 락을 제공한다.

1. MySQL : InnoDB 엔진에서는 공유 잠금(읽기 잠금), 베타적 잠금(읽기 및 쓰기 잠금) 을 제공한다. 개발자는 정책을 알고 잠금의 레벨을 미리 설정한다.
2. Redis : 레디스에서도 lock 기능을 제공하며, `Lettuce`, `Redisson`과 같은 자바 클라이언트에서도 해당 락을 다룰 수 있도록 API를 제공한다.
3. …

참고 : 데드록

정의

2개 이상의 프로세스가 동시에 실행될 때, 다른 작업이 끝나기를 기다리며 동시에 진행되지 못하는 상태이다.

원인은 다른 프로세스와 공유할 수 없는 자원을 사용했거나, 잠금을 사용할 때이다.

조건

자원 또는 프로세스가 다음 4가지 조건을 모두 만족시켰을 때 데드록이 발생한다.

1. 상호배제된 자원 : 다른 프로세스와 공유할 수 없는 배타적 자원이어서 임계구역으로 보호되고 있다.
2. 비선점 자원 : 다른 프로ㅔㅅ스가 이미 사용중인 자원을 빼앗을 수 없어서 대기하고 있다.
3. 점유 및 대기중인 프로세스 : 프로세스가 이미 어떤 자원을 할당받은 상태에서 다른 자원을 대기하고 있다.
4. 원형 대기중인 프로세스 : 대기중인 프로세스들의 관계가 원을 이루고 있다.

해결

데드록은 정책상 오류가 없어도 자연적으로 발생하므로 강압적으로 해결해야 한다. 방법은 다음 3가지가 있다.

1. 예방 : 위 4가지 조건을 무력화 한다. 하지만 현실적으로 불가능하다.
    a. 상호배제하지 않도록 자원을 설정한다.
    b. 모든 자원을 뺏을 수 있게 만든다.
    c. 프로세스가 사용할 자원을 전부 할당하거나 할당하지 않는다.
    d. 자원을 한 방향으로만 사용하게 한다. (유연성이 떨어진다!)
2. 회피 : 자원 할당량을 미리 조절한다.
    - 은행원 알고리즘으로, 각 프로세스의 기대자원이 전체 프로세스의 가용자원보다 크면 자원 할당을 중지하고 지켜본다.
3. 검출 및 회복 : 모니터링하여 데드록 발생을 검출하고, 프로세스 강제종료하여 회복한다.
    a. 검출 : 타임아웃 - 일정시간동안 작업이 진행되지 않는 프로세스를 데드록으로 간주 - 이용하거나, 자원 할당 그래프의 유지/갱신/사이클 검사 추가작업으로 데드록 상태인 프로세스들을 검출한다.
    b. 회복 : 해당 프로세스들을 동시 종료하여, 순차적으로 실핸한다.