22.12.27

📌Concurrency

경쟁, 협력하는 프로세스 간에 생긴 문제점

🌱Process inferaction

1) mulitasking/multiprograming에 의해 프로세스 실행이 interleaving된다(context switching)

2) 다른 프로세스의 활동/ OS가 interrupt를 어떻게 처리할지/ 우선순위 정책에 따라 달라지기 때문에 프로세스의 실행속도는 알수 없다.

3) 이로 인해 발생하는 문제들

- Mutual exclusion
- Deadlock
- Starvation
- Race condition

🌱경쟁에 의한 문제

Lock 사용하여 접근 제어

1) Mutual exclusion

- 한 프로그램(critical section : 자원을 사용하는 곳)에 한 프로세스만 들어가도록 하는 것 = 한 자원에 한 프로세스만 사용한다
- interrupt를 사용하여 critical section에 들어가면 disable, 나올때 enable 표시를 한다.
- 문제
  - CPU를 사용제한한다. → 성능 저하 발생
  - Busy waiting

원하는 자원을 얻기 위해 기다리는 것이 아닌 권한을 얻을 때까지 계속 확인하는 것
→ CPU 자원 낭비, 좋지 않는 동기화 방식

2) Dead Lock

- 2개의 프로세스가 서로 상대방이 가지고 있는 자원을 기다리는 것

3) Starvation

- 무한정 대기
- scheduling 알고리즘에 의해 발생한다.

🌱협력

race condition

- global 변수에 의한 문제(sharing), 공유에 의한 문제
- 발생할 확률이 적지만, 계산한 값이 다르게 나오는 문제 발생
- 2개 이상의 프로세스/쓰레드가 Shared data를 동시에 접근하는데 원하지 않는 결과가 나온 경우, 쓰레드와 프로세스간의 switching 부분에서 어떤것을 먼저 접근할 것이지에 대해서는 알 수 없다. → racing한다.
- critical section으로 보호하여 다른 프로세스가 접근하지 못하도록 한다.

🌱Atomic operation

- critical section을 구현하기 위해서 보통의 프로세서는 atomic한 명령어를 사용한다
- critical section을 보호하기 위한 방법 : lock 변수, 세마포어 변수
- 자원마다 여러개의 critical section을 가진다.

1. atomic 명령어

1) compare and swap instruction

- 비동기화를 위해 멀티쓰레딩에서 사용하는 atomic 명령어
- 특정 메모리 위치값이 주어진 값과 동일하면, 해당 메모리 주소를 새로운 값으로 대체한다.
- 값을 변경하기 위해 한번 더 확인하는 역할

2) exchange instruction

메모리값을 레지스터 값으로 바꾼다.

2. 세마포어

공통된 자원에 접근을 통제하는 변수

분류 1)

- V operation (=signal)

  자원 반환, 세마포어 증가된다.
- P operation(=wait)

  자원요청, 세마포어 감소된다.
- 초기화

분류2)

- Binary : 0과 1만 존재한다.
- Counting

분류3

어떤 프로세스부터 대기 큐(block queue)에서 꺼낼 것인지 기준

- Strong : 선입선출로 꺼낸다.
- Week : 랜덤