Ch6. Synchronization

Created	@Jun 4, 2020 10:45 PM
Tags	운영체제

프로세스는 동시에 실행될 수 있으며, 여러 개의 프로세스가 협력할 때는 프로세스 사이에 데이터가 동기화되지 않는 문제가 발생할 수 있다.

Background

만약 두 프로세스가 동시에 어떤 변수의 값을 바꾼다면 프로그래머의 의도와는 다른 결과가 나올 것이다. 이처럼 **프로세스가 어떤 순서로 데이터에 접근하느냐에 따라 결과 값이 달라질 수 있는 상황을 경쟁 상태**(race condition)라고 한다.

The Critical-Section Problem

코드상에서 경쟁 조건이 발생할 수 있는 특정 부분을 critical section이라고 부른다. critical section problem를 해결하기 위해서는 몇가지 조건을 충족해야 한다.

- Mutual exclution (상호 배제): 이미 한 프로세스가 critical section에서 작업중일 때 다른 프로세스는 critical section에 진입해서는 안 된다.
- Progress (진행): critical section에서 작업중인 프로세스가 없다면 다른 프로세스가 critical section에 진입할 수 있어야 한다.
- Bounded waiting (한정 대기): critical section에 진입하려는 프로세스가 무한하게 대기해서는 안 된다.

Non-preemptive kernels로 구현하면 임계 영역 문제가 발생하지 않는다. 하지만 비선점 스케줄링은 반응성이 떨어지기 때문에 슈퍼 컴퓨터가 아니고선 잘 사용하지 않는다.

Peterson's Solution

Peterson's solution으로 임계 영역 문제를 해결할 수 있다. 임계 영역에서 **프로세스가 작업중인지 저장하는 변수** `flag`와 **critical section에 진입하고자하는 프로세스를 가리키는 변수** `turn` 을 만들어 어떤 프로세스가 임계 영역에 진입하면 `flag`를 `lock`하고, 나오면 `unlock`하는 방식으로 임계 영역 문제를 해결한다.

```
do {
  flag[i] = true;
  turn = j;
  while (flag[j] && turn == j);
  // Critical section
  flag[i] = false;
  // Remainder section
} while(true);
```

Mutex Locks

mutex locks은 여러 스레드가 공통 리소스에 접근하는 것을 제어하는 기법으로, `lock`이 하나만 존재할 수 있는 locking 매커니즘을 따른다. (참고로 'mutex'는 'MUTual EXclusion'을 줄인 말이다.) **이미 하나의 스레드가 critical section에서 작업중인 `lock` 상태에서 다른 스레드들은 critical section에 진입할 수 없도록 한다.**

Semaphores

세마포어(Semaphore)는 여러 개의 프로세스나 스레드가 critical section에 진입할 수 있는 locking 매커니즘이다. **세마포어는 카운터를 이용해 동시에 리소스에 접근할 수 있는 프로세스를 제한한다.** 물론 한 프로세스가 값을 변경할 때 다른 프로세스가 동시에 값을 변경하지는 못한다. 세마포어는 P와 V라는 명령으로 접근할 수 있다. (P, V는 try와 increment를 뜻하는 네덜란드어 Proberen과 Verhogen의 머릿글자다.)

Semaphore Usage

세마포어의 카운터가 한 개인 경우 바이너리 세마포어(Binary semaphore), 두 개 이상인 경우 카운팅 세마포어(Counting semaphore)라고 한다. 바이너리 세마포어는 사실상 mutex와 같다.

Deadlocks and Starvation

두 프로세스가 서로 종료될 때까지 대기하는 프로그램을 실행한다고 생각해보자. 프로세스 A는 B가 종료될 때까지, 프로세스 B는 A가 종료될 때까지 작업을 하지 않기 때문에 프로그램은 어떤 동작도 하지 못할 것이다. 이처럼 **두 프로세스가 리소스를 점유하고 놓아주지 않거나, 어떠한 프로세스도 리소스를 점유하지 못하는 상태가 되어 프로그램이 멈추는 현상을 데드락(Deadlock)이라고 한다.** 운영체제도 결국 소프트웨어이기 때문에 데드락에 빠질 수 있다.

Classic Problems of Synchronization

데드락에 관한 유명한 비유가 있다. 철학자 5명이 식탁 가운데 음식을 두고 철학자들은 사색과 식사를 반복한다. 포크는 총 5개, 단 음식을 먹으려면 2개의 포크를 사용해야 한다. 즉, 동시에 음식을 먹을 수 있는 사람은 두 명뿐이다. 운이 좋으면 5명의 철학자들이 돌아가면서 사색과 식사를 이어갈 수 있다. 하지만 **모두가 포크를 하나씩 들고 식사를 하려하면 누구도 식사를 할 수 없는 상태**, 다시말해 데드락에 빠져 버린다. 이것이 바로 철학자들의 만찬 문제(Dining-Philosophers Problem)이다.