**사전 지식 정리 – 공유 메모리(Shared Memory)**

* 컴퓨터 환경에서 여러 프로그램이 동시에 접근할 수 있는 메모리이다. 과다한 복사를 피하거나 해당 프로그램 간 통신을 위해 고안되었다.
* 환경에 따라 프로그램은 하나의 프로세서에서나 여러 개의 프로세서에서 실행할 수 있다.
* (예를 들어 여러 개의 스레드 간에) 하나의 프로그램 안에서 통신을 위해 메모리를 사용하는 일은 일반적으로 공유 메모리로 부리지 않는다.

1. 하드웨어에서

: 컴퓨터 하드웨어에서 공유 메모리는 멀티프로세서 컴퓨터 시스템에서 각기 다른 여러 중앙 처리 장치(CPU)가 접근할 수 있는, 일반적으로 큰 랜덤 액세스 메모리(RAM)을 가리킨다.

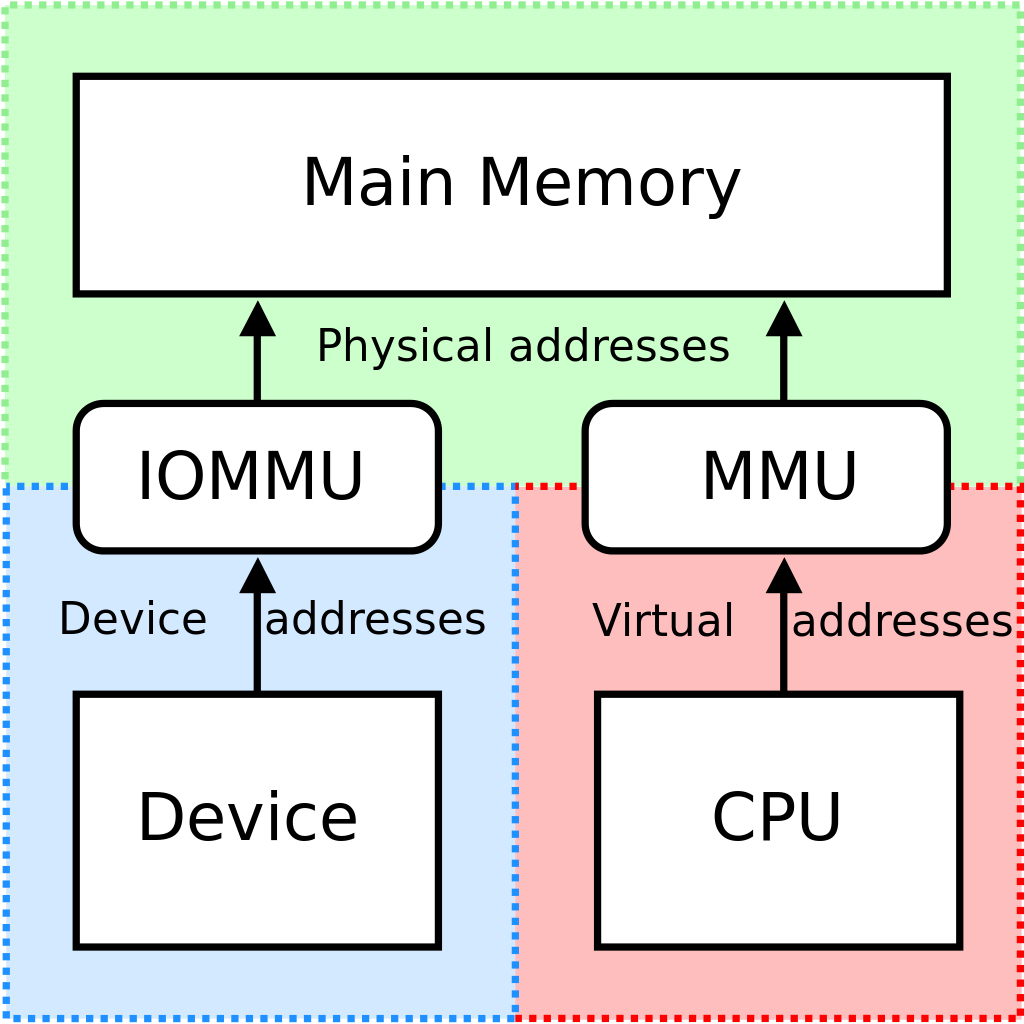
공유 메모리는 다음을 사용할 수 있다.

1. 균일 기억 접근 장치(UMA)
2. 불균일 기억 장치 접근(NUMA)
3. 캐시 전용 메모리 구조(COMA)
4. 소프트웨어에서

: 컴퓨터 소프트웨어에서 공유 메모리는 다음 중 하나이다.

프로세스 간 통신(IPC) 방식

가상 메모리 매핑을 사용하거나 프로그램의 명시적 지원이 있을 때 하나의 인스턴스에 대한 데이터 사본으로의 접근을 지시하면서 메모리 공간을 보존하는 방식.



* 공유메모리는 여러 IPC 중에서 가장 빠른 수행속도를 보여준다.

(IPC (inter-Process Communication)

: 프로세스 간 통신, 프로세스들 사이에 서로 데이터를 주고받는 행위 또는 그에 대한 방법이나 경로)

그 이유는 하나의 메모리를 공유해서 접근하게 되므로, 데이터 복사와 같은 불필요한 오버헤드가 발생하지 않기 때문

빠른 데이터의 이용이 가능하다. 그러나 하나의 프로세스가 메모리에 접근 중에 있을 때, 또 다른 프로세스가 메모리에 접근하는 일이 발생하면 자칫 데이터가 훼손될 수 있으므로 한번에 하나의 프로세스가 메모리에 접근하고 있다는 것을 보증해 주어야 한다.

**세마포어와 뮤텍스**

1. 세마포어(Semaphore)

: 공유된 자원의 데이터를 여러 프로세스가 접근하는 것을 막는 것

1. 뮤텍스(Mutex)

: 공유된 자원의 데이터를 여러 쓰레드가 접근하는 것을 막는 것.

1. Critical Section(임계 영역)

: 다중 프로그래밍 운영체제에서 여러 프로세스가 데이터를 공유하면서 수행될 때 각 프로세스에서 공유 데이터를 액세스하는 프로그램 코드 부분을 가리키는 말.

공유 데이터를 여러 프로세스가 동시에 액세스하면 시간적인 차이 때문에 잘못된 결과를 만들어 낼 수 있기 때문에 한 프로세스가 위험 부분을 수행하고 있을 때, 즉 공유 데이터를 액세스하고 있을 때는 다른 프로세스들은 절대로 그 데이터를 액세스하지 못하도록 하여야 한다.

예제 )

컴퓨터가 여러 프로그램을 동시에 수행하는 다중 프로그래밍 시스템에서는 프로세스들간의 상호배제와 동기화를 위한 기본적인 연산이 필요하게 되고 세마포어는 여러 프로세스들에 의해 공유되는 변수로 정의됩니다.

그런데 이 변수는 보통의 방법으로는 액세스할 수 없고 오직 P와 V라는 연산으로만 다룰 수 있습니다.

P와 V연산의 정의는 아래와 같습니다.

-------------------------

procedure P(S)            --> 최초 S값은 1

while S=0 do wait        --> S가 0이면 1이 될 때까지 wait

S := S-1                     --> S를 0로 만들어 다른 프로세스가 들어 오지 못하도록 함

end P

-------------------------

-------------------------

procedure V(S)          --> 현재상태는 S가 0

S := S+1                  --> S를 1로 원위치시켜 해제하는 과정

end V                     -->이제는 다른 프로세스가 들어 올수 있음

-------------------------

즉 한 프로세스가 P나 V를 수행하고 있는 동안에는 프로세스가 인터럽트를 당하지 않게 됩니다. 이제 P와 V를 사용하면 다음과 같이 위험지역(cirtical section)에 대한 상호배제를 구현할 수 있게 됩니다.P(S);

------------------------

위 험 지 역(Critical Section) = 임계영역

------------------------

V(S);

최초에 S의 값은 1이고, 위와 같은 위험지역을 포함하는 두개의 프로세스 A와 B가 있다고 할 때,

A와 B는 서로 독립적으로 수행되지만, 두 프로세스가 동시에 위험 지역으로 들어가서는 안된다.

위와 같이 세마포어를 사용하면 P(S)를 먼저 수행하는 프로세스가 S를 0으로 해놓고 위험지역에 들어가므로 나중에 도착하는 프로세스는 P에서 더이상 진행되지 못하고 기다리게 된다. 먼저 들어갔던 프로세스가 V(S)를 해주어야 비로서 P(S)에서 기다리던 프로세스가 위험지역에 들어갈 수 있고 따라서 상호배제가 실현된다.

**정리**

* 뮤텍스(Mutex)란?
* Mutual Exclusion 으로 상호배제라고도 한다.
* Critical Section을 가진 스레드들의 Running Time이 서로 겹치지 않게 각각 단독으로 실행되게 하는 기술.
* 다중 프로세스들의 공유 리소스에 대한 접근을 조율하기 위해 locking과 unlocking을 사용한다.
* 즉, 쉽게 말하면 뮤텍스 객체를 두 스레드가 동시에 사용할 수 없다는 의미.
* 세마포어란(Semaphore)
* 세마포어는 리소스의 상태를 나타내는 간단한 카운터로 생각할 수 있다.
* 일반적으로 비교적 긴 시간을 확보하는 리소스에 대해 이용하게 된다.
* 유닉스 시스템의 프로그래밍에서 세마포어는 운영체제의 리소스를 경쟁적으로 사용하는 다중 프로세스에서 행동을 조정하거나 또는 동기화 시키는 기술
* 세마포어는 운영체제 또는 커널의 한 지정된 저장장치 내 값으로서, 각 프로세스는 이를 확인하고 변경할 수 있다.
* 확인되는 세마포어의 값에 따라, 그 프로세스가 즉시 자원을 사용할 수 있거나 또는 이미 다른 프로세스에 의해 사용 중이라는 사실을 알게 되면 재시도하기 전에 일정 시간을 기다려야만 한다.
* 세마포어는 이진수 (0 또는 1)를 사용하거나 또는 추가적인 값을 가질 수 있다.
* 세마포어를 사용하는 프로세스는 그 값을 확인하고, 자원을 사용하는 동안에는 그 값을 변경함으로써 다른 세마포어 사용자들이 기다리도록 한다.
* Mutex vs Semaphore
* 세마포어는 뮤텍스가 될 수 있지만, 뮤텍스는 세마포어가 될 수 없다.

(Mutex는 상태가 0,1 두 개 뿐인 binary Semaphore)

* 세마포어는 소유할 수 없는 반면, 뮤텍스는 소유가 가능하며 소유주가 이에 대한 책임을 진다.

(뮤텍스의 경우 상태가 두 개 뿐인 lock이므로 lock을 ‘가질’ 수 있다.)

* 뮤텍스의 경우 뮤텍스를 소유하고 있는 스레드가 이 뮤텍스를 해제할 수 있다. 하지만 세마포어의 경우 이러한 세마포어를 소유하지 않는 스레드가 세마포어를 해제할 수 있다.
* 세마포어는 시스템 범위에 걸쳐있고 파일 시스템상의 파일 형태로 존재한다. 반면 뮤텍스는 프로세스 범위를 가지며 프로세스가 종료될 때 자동으로 Clean up된다.
* 가장 큰 차이점
* 관리하는 동기화 대상의 개수
* 뮤텍스는 동기화 대상이 오직 하나 뿐일 때, 세마포어는 동기화 대상이 하나 이상일 때 사용한다.