**사전 지식 정리\_세마포어, 데드락**

1. 세마포어(Semaphore)

* 두 개의 원자적 함수로 조작되는 정수 변수로서, 멀티프로그래밍 환경에서 공유 자원에 대한 접근을 제한하는 방법으로 사용된다.
* 교착 상태를 해결하지는 못한다.
* 구성

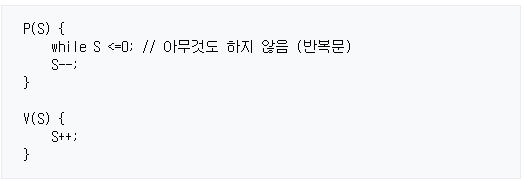
: S는 정수값을 가지는 변수이며, 다음과 같이 P와 V라는 명령에 의해서만 접근할 수 있다.

P는 임계 구역에 들어가기 전에 수행되고, V는 임계 구역에서 나올 때 수행된다. 이 때 변수 값을 수정하는 연산은 모두 원자성을 만족해야 한다.

다시 말해, 한 프로세스(또는 스레드)에서 세마포어 값을 변경하는 동안 다른 프로세스가 동시에 이 값을 변경해서는 안된다.

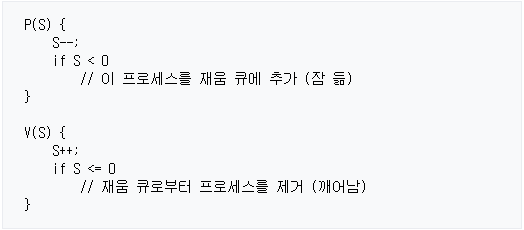
1. 방법 1

: 최초 제시된 방법은 바쁜 대기(busy waiting)을 이용한 방법.



1. 방법 2

: 최초 방법의 단점을 보완한 방법으로서 재움 큐를 활용하여 프로세스를 재우는 방식.



* 종류

1. 계수 세마포어

: 초기값은 가능한 자원의 수로 정해지며, 세마포어 값의 범위는 정해져 있다.

1. 이진 세마포어

: 세마포어 값으로 0 또는 1을 가진다. 계수 세마포어보다 간단히 구현할 수 있으며, Test and Set 등 하드웨어가 지원하는 기능을 이용하여 구현하기도 한다. 또한, 이진 세마포어를 이용하여 계수 세마포어를 구현할 수 있다.

* 약점

: P함수와 V함수의 동작은 독립적이기 때문에 잘못 사용하는 경우 문제가 발생한다.

P – 임계구형, 현재 프로세스가 임계 구역에서 빠져나갈 수 없게 된다. 또한 다른 프로세스들은 임계 구역에 들어갈 수 없으므로 교착상태(Deadlock)가 발생한다.

V – 임계 구역 : 2개 이상의 프로세스가 동시에 임계구역에 들어갈 수 있으므로 상호 배제(Mutual Exclusion)를 보장할 수 없게 된다.

고급 언어에서 동기화를 제공해야 한다.

교착 상태

* 두 개 이상의 작업이 서로 상대방의 작업이 끝나기 만을 기다리고 있기 때문에 결과적으로 아무것도 완료되지 못하는 상태를 가리킨다.
* 교착 상태의 조건

1. 상호배제(Mutual exclusion)

: 프로세스들이 필요로 하는 자원에 대해 배타적인 통제권을 요구한다.

1. 점유대기(Hold and wait)

: 프로세스가 할당된 자원을 가진 상태에서 다른 자원을 기다린다.

1. 비선점(No preemption)

: 프로세스가 어떤 자원의 사용을 끝낼 때까지 그 자원을 뺏을 수 없다.

1. 순환대기(Circular wait)

: 각 프로세스는 순환적으로 다음 프로세스가 요구하는 자원을 가지고 있다.

: 이 조건 중에서 한 가지라도 만족하지 않으면 교착 상태는 발생하지 않는다. 이중 순환대기 조건은 점유대기 조건과 비선점 조건을 만족해야 성립하는 조건이므로, 위 4가지 조건은 서로 완전히 독립적인 것은 아니다.

(비선점 스케쥴링 : 프로세스가 자원을 할당 받았을 경우, 자원을 스스로 반납할 때까지 계속 그 자원을 사용하도록 허용하는 정책이다.)

* 비선점 시스템의 스케줄링

1. 우선순위 알고리즘

:각 프로세스에게 우선 순위를 부여하여 순위가 높은 순서대로 처리하는 방법.

1. 기한부(Deadline) 스케줄링

:작업들이 명시된 시간이나 기한 내에 완료되도록 게획. 사전에 정확한 자원 및 수행 시간을 예측하기 어려움

1. FCFS

: CPU 스케줄링 알고리즘 중에 제일 간단한 알고리즘으로 CPU를 요구하는 순으로 할당하는 방법이며 FIFO 방식인 큐로 써서 구현된다.

1. SJF

: 각각의 작업이 다음 CPU버스트(Bust) 시간을 중심으로 CPU가 사용 가능할 때 버스트 시간이 가장 작은 작업에 할당하는 기법이다.

* 교착 상태의 관리

: 현재의 대부분의 운영체제들은 교착 상태를 막는 것은 불가능하다. 교착 상태가 발생하면 여러 운영 체제들은 제각기 다른 비표준 방식들로 이러한 교착 상태에 대응한다.

대부분의 접근들은 4가지 코프먼 조건들 가운데 하나(특히 4번째 것)를 막음으로써 동작한다.

* 교착 상태의 예방

1. 상호배제 조건의 제거

: 교착 상태는 두 개 이상의 프로세스가 공유가능한 자원을 사용할 때 발생하는 것이므로 공유 불가능한, 즉 상호 배제 조건을 제거하면 교착상태를 해결할 수 있다.

1. 점유와 대기 조건의 제거

: 한 프로세스에 수행되기 전에 모든 자원을 할당시키고 나서 점유하지 않을 때에는 다른 프로세스가 자원을 요구하도록 하는 방법이다. 자원 과다 사용으로 인한 효율성, 프로세스가 요구하는 자원을 파악하는 데에 대한 비용, 자원에 대한 내용을 저장 및 복원하기 위한 비용, 기아 상태, 무한 대기 등의 문제점이 있다.

(기아 상태 : 프로세스가 끊임없이 필요한 컴퓨터 자원을 가져오지 못하는 상황)

1. 비선점 조건의 제거

: 비선점 프로세스에 대해 선점 가능한 프로토콜을 만들어 준다.

1. 환형 대기 조건의 제거

: 자원 유형에 따라 순서를 매긴다.

* 이 교착 상태의 해결 방법들은 자원 사용의 효율성이 떨어지고 비용이 많이 드는 문제점이 있다.
* 교착 상태의 회피

: 자원이 어떻게 요청될지에 대한 추가정보를 제공하도록 요구하는 것으로 시스템에 환형대기(Circular Wait)가 발생하지 않도록 자원 할당 상태를 검사한다.

1. 자원 할당 그래프 알고리즘
2. 은행원 알고리즘

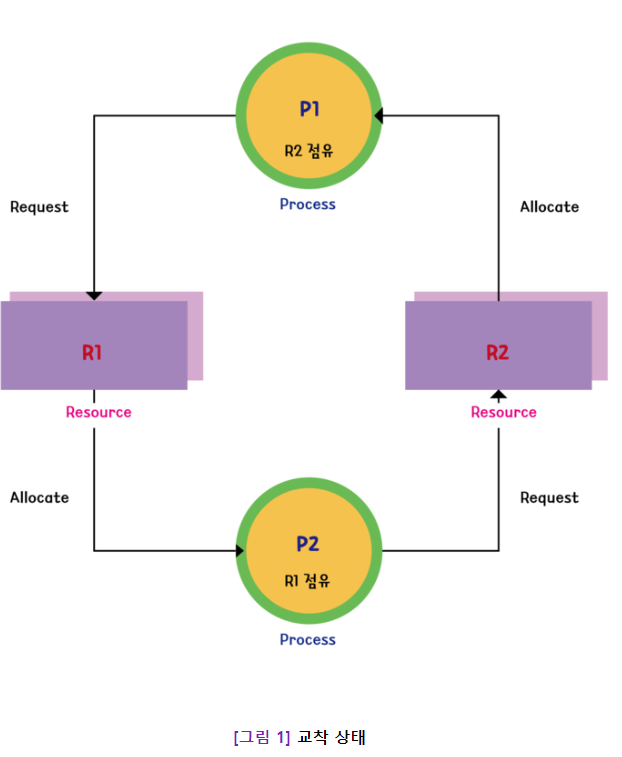
* 교착 상태의 무시

: 예방 혹은 회피기법을 프로그래밍해서 넣으면 성능에 큰 영향을 미칠 수 있게 된다.

그렇기 때문에 Deadlock의 발생 확률이 비교적 낮은 경우 별다른 조치를 취하지 않는다.

* 교착 상태의 발견

: 감시/발견을 하는 detection 알고리즘으로 Deadlock 발생을 체크하는 방식.

이 역시 성능에 큰 영향을 미칠 수 있다.

1. 상호배제(Mutual Exclusion)

: 한 번에 한 프로세스만 해당 자원을 사용할 수 있어야 한다.

Ex) 자원 R1은 P1 or P2 프로세스 중 하나의 프로세스에서만 사용해야 한다.

1. 점유와 대기(Hold and Wait)

: 프로세스가 한 개 이상의 자원을 할당 받은 후 다른 프로세스에 할당된 자원을 기다리는 상태.

Ex) P1 프로세스는 자원 R2를 할당 받은 후 P2 프로세스에 할당된 자원을 얻기 위해 기다리는 중

1. 비선점(No Preemption)

: 다른 프로세스가 사용중인 자원을 강제로 뺏을 수 없다.

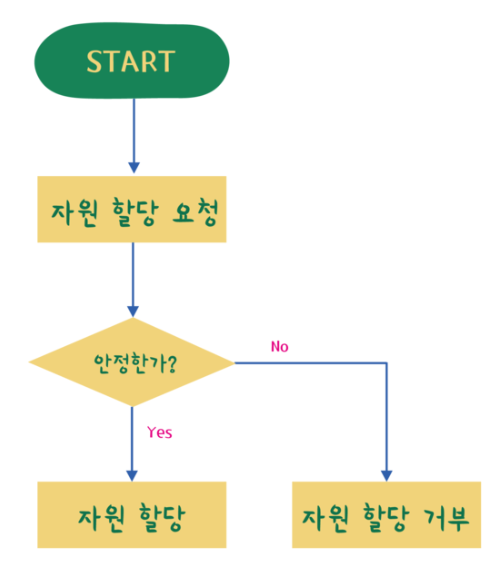
1. 순환 대기(Circular Wait)

: 프로세스들이 순환을 이루어서 존재, 이를 구성하는 각 프로세스는 이전 프로세스가 요청하는 자원을 점유하고 다음 프로세스가 점유하고 있는 자원을 요청하는 경우

점유대기, 비선점, 상호배제 조건을 만족해야 발생

Ex) P1 프로세스는 P2 프로세스가 요청한 자원 R2를 점유하고 다음 프로세스인 P2가 점유하고 있는 자원 R1을 요청하고 있다. P2 프로세스도 마찬가지.

-Banker’s Algorithm



* 프로세스가 자원을 요구할 때 시스템이 자원을 할당 한 후 안정 상태로 남아있게 되는지 사전에 검사하여 교착상태를 회피하는 방법.
* 여러 유형의 자원을 가진 시스템에서 사용되며, 시스템이 자원을 할당한 후에도 안정 상태로 남아있는지 검사하여 안정 상태에 있으면 자원을 할당하고 그렇지 않으면 다른 프로세스들이 자원을 해제할 때까지 대기 합니다.
* 단점

1. 할당할 수 있는 자원의 수가 일정해야 한다.
2. 사용자 수가 일정해야 한다.
3. 항상 불안정 상태를 방지해야 하므로 자원 이용도가 낮다.

