교착상태 (Deadlock)

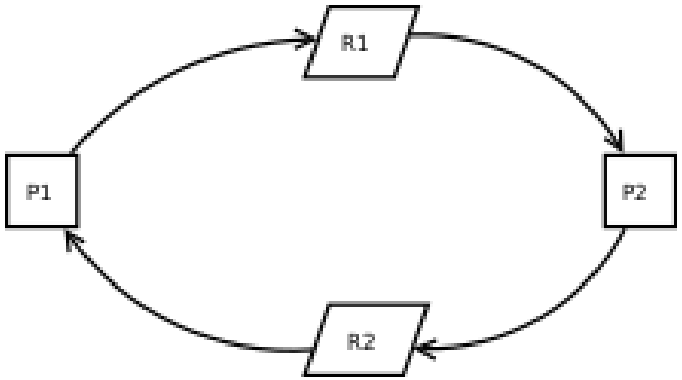
Computer Science - Operaitng System

목차

- 교착상태란?
- 교착상태가 발생하는 조건
  - 상호배제 (Mutual exclusion)
  - 점유대기 (Hold and wait)
  - 비선점 (No preemption)
  - 순환대기 (Circular wait)
- 교착상태 관리
  - 교착상태 예방
  - 교착상태 회피
  - 교착상태 탐지 및 회복

교착상태란?

- 두 개 이상의 프로세스들이 서로가 가진 자원을 기다리며 중단된 상태를 말한다.



* 프로세스는 지정된 태스크를 수행하기 위해 필요한만큼의 자원을 요청할 수 있다. (시스템에서 상요 가능한 전체 자원수 이내로)
* 프로세스는 자원을 사용하기 전에 반드시 요청해야 하고, 사용 후에는 반드시 방출해야 한다.

1. **요청** : 요청이 즉시 허용되지 않으면 요청 프로세스는 자원을 얻을 때까지 대기
2. **사용** : 프로세스는 자원에 대해 작업을 수행
3. **방출** : 프로세스가 자원을 방출

(ex. 장치-request()/resoponse(), 파일-open()/close(), 메모리-allocate()/free(), 세마포어-wait(), signal() 연산 등)

교착상태가 발생하는 조건

교착상태는 한 시스템에 다음 네가지 조건이 동시에 성립될 때 발생

- **상호배제** (*Mutual exclusion*)
  자원은 하나의 프로세스만 사용할 수 있다.
- **점유대기** (*Hold and wait*)
  프로세스가 할당된 자원을 가진 상태에서 다른 자원을 기다린다.
- **비선점** (*No preemption*)
  자원들을 선점할 수 없다. 자원이 강제적으로 방출될 수 없고, 점유하는 프로세스가 태스크를 종료한 후 그 프로세스에 의해서만 방출될 수 있다.
  즉, 프로세스가 어떤 자원의 사용을 끝낼 때 까지 그 자원을 뺏을 수 없다.
- **순환대기** (*Circular wait*)
  각 프로세스는 순환적으로 다음 프로세스가 요구하는 자원을 가지고 있다.

교착상태 관리 - 예방

교착상태를 관리하기 위해서는 교착상태 조건을 충족시키지 않도록 하는 예방과 회피하는 방법, 발생확률이 낮은 교착상태를 무시하는 방법이 있다.

- 교착상태 예방
  교착상태 발생 조건 중 한 개라도 충족시키지 않게 하여 교착상태를 예방하는 방법.
  하지만 이는 장치의 이용률이 저하되고, 시스템 처리율이 감소된다.

  - **상호배제 조건 제거**
    여러 프로세스가 공유자원 사용할 수 있도록 한다.
  - **점유대기 조건 제거**
    프로세스 실행 전 모든 자원을 할당, 점유하지 않을 때 다른 프로세스가 자원을 요구할 수 있다.
  - **비선점 조건 제거**
    비선점 프로세스에 대해 선점 가능한 프로토콜을 만든다.
    자원 할당을 요청하면 바로 선점되어야 하기 때문에 CPU 레지스터나 메모리같이 중간 저장 및 복원이 용이한 곳에서 사용 되며, 실제로 교착상태가 잘 일어나지 않는다.
  - **환형 대기 조건 제거**
    자원 유형에 따라 순서를 매긴다.
    모든 자원유형에 전체적인 순서를 부여 후 각 프로세스가 열거된 순서(오름차순)대로 자원을 요청하도록 요구한다.

교착상태 관리 - 회피

- Safe state
  시스템의 프로세스들이 요청하는 모든 자원을 교착상태를 발생시키지 않고 차례로 모두 할당할 수 있는 상태
- Safe sequence
  Safe State에 도달할 수 있도록 하는 (교착상태가 발생하지 않도록) 작업 순서
- Unsafe state
  Safe state를 찾을 수 없는 상태. Unsafe state라고 해서 교착상태가 일어나는 것은 아니다. 하지만 '교착상태로 갈 수 있는 상태'. Safe state를 유지하면 교착상태와 Unsafe state를 모두 예방할 수 있다.

- 회피 알고리즘의 기본은 시스템의 상태가 Safe state를 유지하도록 하는 것

교착상태 관리 - 회피

- **자원 할당 그래프 알고리즘**

  자원별로 하나의 인스턴스만 가질 때 사용할 수 있는 알고리즘으로, 요청 간선, 할당 간선으로 이루어진 자원할당 그래프에 예약 간선(미래에 자원 R을 요청할 것이라는 의미)을 추가하여 예약 간선간 그래프의 사이클이 생기지 않을 때만 자원 할당을 허용하도록 하여 교착상태를 방지한다.

- **은행원 알고리즘**

  자원 별 여러 인스턴스가 존재할 때 사용할 수 있는 알고리즘. 자원 총 보유 수, 각 프로세스 별 최대 자원 요청량(Max), 현재 프로세스에 할당 중인 자원 양(Allocation), 남은 필요한 자원양의 정보(Need)를 이용하여 가용가능한 자원양(Available) 정보를 얻고 Safe sequence를 도출한다.

  예제) 시스템에는 A자원 10개, B자원 5개, C자원 7개가 있음.

(t=t0)	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

교착상태 관리 - 회피

1. 각 프로세스 별 **Need 행렬**을 구한다. (Max –Allocation)

(t=t0)	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Max-Alloc

(t=t0)	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	4	3	3	3	2
P1	2	0	0	1	2	2			
P2	3	0	2	6	0	0			
P3	2	1	1	0	1	1			
P4	0	0	2	4	3	1			

2. 프로세스마다 필요한 자원(Need) 정보와 할당 가능한 자원(Available) 정보를 비교하여 프로세스의 자원할당 순서를 정해 Safe Sequence를 완성한다. (안정성 알고리즘, 자원요청 알고리즘 포함)
   1. P0 : N (7, 4, 3) > A (3, 3, 2) (X), **Safe Seq :**
   2. P1 : N (1, 2, 2) < A (3, 3, 2) (O), 자원해제 후 Available : (5, 3, 2), **Safe Seq : P1**
   3. P2 : N (6, 0, 0) > A (5, 3, 2) (x), **Safe Seq : P1**
   4. P3 : N (0, 1, 1) < A (5, 3, 2) (O), 자원해제 후 Available : (7, 4, 3), **Safe Seq : P1 -> P3**
   5. P4 : N (4, 3, 1) < A (7, 4, 3) (O), 자원해제 후 Available : (7, 4, 5), **Safe Seq : P1 -> P3 -> P4**
   6. P0 : N (7, 4, 3) < A (7, 4, 5) (O), 자원해제 후 Available : (7, 5, 5), **Safe Seq : P1 -> P3 -> P4 -> P0**
   7. P2 : N (6, 0, 0) < A (7, 5, 5) (O), 자원해제 후 Available : (10, 5, 7), **Safe Seq : P1 -> P3 -> P4 -> P0 -> P2**

교착상태 관리 – 탐지 및 회복

- **탐지**
  Allocation, Request, Available 등으로 시스템에 데드락이 발생했는지 여부를 탐색한다. 은행원 알고리즘에서 했던 방식과 유사하게 현재 시스템의 자원 할당 상태를 가지고 파악.
  이 외에도 자원 할당 그래프를 통해 탐지 가능.
- **회복**
  데드락을 탐지 기법을 통해 발견 후, 순환 대기에서 벗어나 데드락으로부터 회복하기 위한 방법을 사용한다.
  - **단순히 프로세스를 1개 이상 중단시키기**
    - 교착 상태에 빠진 모든 프로세스를 중단시키는 방법
    - 프로세스를 하나씩 중단 시킬 때마다 탐지 알고리즘으로 데드락을 탐지하면서 회복시키는 방법
  - **자원 선점하기**
    - 교착 상태의 프로세스가 점유하고 있는 자원을 선점해 다른 대기중인 프로세스에 할당 (해당 프로세스 일시정지)
    - 자원 선점을 위해 희생자 선택, 롤백, 기아상태에 대해 고려해야 한다.