**교착상태-deadlock**

**Blocked/asleep state**- 프로세스가 기다리는 상태 자원을 기다리는 상태

**Deadlock state**- (asleep 상태에 존재) 프로세스가 발생 가능성이 없는 이벤트를 기다리는 경우

**Starvation**- (ready 상태에 존재) 운이 없어서 우선순위에 밀려서 기다리고 있는 상태

**자원의 분류**

**일반적 분류**

Hardware resource vs software resources

**선점 가능 여부에 따른 분류** –pre-emptible resources 선점 당한 후 돌아와도 문제가 발생하지 않는 자원 (processor, memory)

-non-pre-emptible resources 선점 당하면 이후 진행에 문제가 발생하는 자원 (disk drive)

**할당 단위에 따른 분류** – total allocation resources –partitioned allocation resources

**동시에 사용 가능 여부에 따른 분류** -exclusive allocation resources –shared allocation resources

**재사용 가능 여부에 따른 분류** – (SR) serially-reusable resources –CR (consumable resources)

**Deadlock 을 발생 시킬 수 있는 자원**- non-pre-emptible resources, exclusive allocation resources, serially reusable resources

**Deadlock Model**

1. graph model: node + edge

2. State Transition model – 프로세스는 한번에 하나만 요청 반납 가능

**Deadlock 발생 필요 요건**

-exclusive use of resources, non-pre-emptible resources, hold and wait, circular wait

**Deadlock prevention**: 4개의 deadlock 발생 요건 중 하나를 제거

* 모든 자원을 공유 허용: 불가능
* 모든 자원에 대해 선점 허용: 현실적으로 불가능, 비현실성
* 필요 자원 한번에 모두 할당: 자원 낭비 발생, 무한 대기 발생 가능
* Circular wait 조건 제거: 자원에게 순서를 부여

**Deadlock avoidance**: 시스템 상태를 감시, 자원 할당 요청 보류, safe state로 유지, 최악의 경우

* Safe state –모든 프로세스가 정상적으로 종료 가능한 상태
* Unsafe state –deadlock 상태가 될 가능성이 있음
* Dijkstra’s banker’s algorithm(한가지 자원)
* habermann’s algorithm(여러가지 자원 고려)
* high overhead (항상감시)
* low resources utilization
* not practical

**Deadlock detection and Recovery** – 주기적으로 deadlock 확인, 최선의 경우를 생각

* Resource allocation graph (RAG)
* Graph reduction: 1. unblocked process- 필요한 자원을 다 할당 받을 수 있는 프로세스를 모두 지운다

2. completed reduced – 모든 edge가 제거- deadlock이 없음

3. irreducible – edge가 남음, deadlock이 존재

4. 검사 주기에 영향을 받음

- Deadlock recovery: 1. Process termination- deadlock 프로세스 중 일부 종류

2. Resource preemption – 선점할 자원을 선택

3. cost model이 필요

4. checkpoint-restart method -rollback