

과정명	
01차시	운영체제의 관리

<1> 프로세스 관리

[1] 프로세스의 종류

- (1) 운영체제 프로세스
 - 프로세스의 실행 순서 제어, 시스템/응용 프로그램 감시, 사용자나 입출력 프로세스 생성
- (2) 사용자 프로세스
 - 사용자 코드를 실행(사용자가 실행하는 프로세스)
- (3) 병행 프로세스
 - 여러 개의 프로세스가 동시에 실행되는 것
 - 동시에 시스템에 존재하지만, 어느 순간에는 한 프로세스만 실행
 - 독립적으로 실행되거나 다른 프로세스와 협력하여 실행됨
 - 두 개 이상의 프로세스가 병행 처리 상태에 있으면 예측 불가능한 결과가 발생할 수 있음
이런 오류 방지를 위해 동기화, 상호배제, 임계구역 기법을 사용함

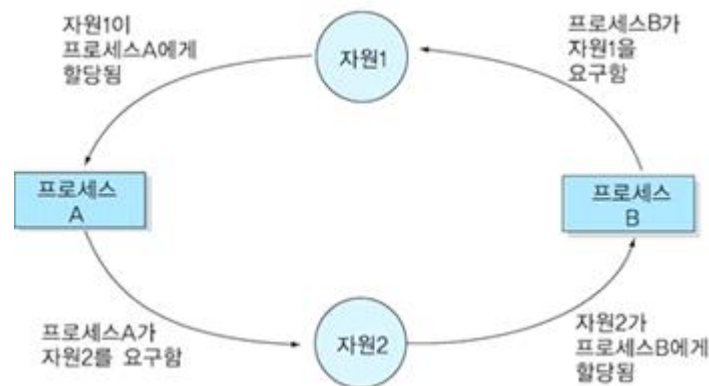
[2] 프로세스의 상태 전이

- 컴퓨터 시스템 내에 존재하는 프로세스는 생성 후 종료될 때까지 다양한 사건에 의해 접수, 준비, 실행, 대기, 종료 상태 등 여러 가지 상태 변화를 거침



[3] 프로세스의 교착상태

- 다중 프로그래밍 시스템에서 하나 이상의 프로세스가 절대 일어나지 않을 특정 사건을 기다리고 있는 상태



(1) 교착상태의 발생 조건

- 교착상태는 상호 배제, 보유와 대기, 비선점, 환형 대기의 조건이 만족될 때 발생함
- 상호 배제는 오직 하나의 프로세스만이 자원을 사용할 수 있다는 것으로, 다른 프로세스가

사용 중인 그 자원을 반환될 때까지 기다려야 함

- 보유와 대기는 어떤 프로세스가 자신에게 할당된 자원을 보유하면서 다른 프로세스가 보유하고 있는 자원을 추가로 요구하는 것으로 해당 자원이 반환될 때까지 기다리는 상태임
- 비선점은 어떤 프로세스가 할당된 자원을 다른 프로세스가 강제로 선점할 수 없다는 것으로, 프로세스가 할당된 자원을 사용한 후 반환하기 전에는 회선할 수 없음
- 환형 대기는 각 프로세스가 자신에게 할당된 자원을 보유하면서 상대방의 자원을 요청하는 상태로 프로세스와 자원들이 원형을 이루게 됨

(2) 교착상태의 해결 방안

- 교착상태의 해결방안은 예방, 회피, 탐지, 회복의 4가지로 분류할 수 있음
- 교착상태 예방은 발생 조건 4가지 중 하나를 제거하는 것으로 분명한 해결책이지만 정확한 자원 사용 정책을 제시해야 하므로 자원이 낭비될 수 있음
- 교착상태 회피는 시스템 운영 중 상황에 따라 교착 상태 발생 가능성을 피해가는 방법임
- 교착상태 탐지는 일단 교착상태가 발생하도록 허용한 후, 시스템 운영 중 교착 상태의 발생 여부를 판단하는 방법으로 교착상태를 탐지 후 관련 프로세스와 자원을 결정함
- 교착상태 회복은 교착상태 탐지 후 교착 상태에 관련된 하나 이상의 프로세스를 시스템에서 제거하고, 할당된 자원을 다른 프로세스에 제공하여 교착상태로부터 회복하는 방법임

<2> CPU 스케줄링

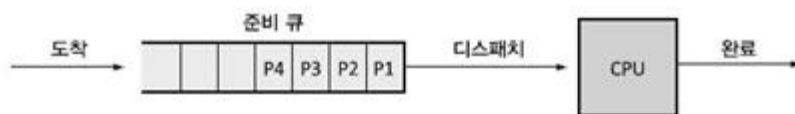
[1] 스케줄링 성능 기준

- (1) CPU 활용률(CPU 총 작동 시간 대비 실제 사용 시간)
- (2) 처리율(단위 시간당 완료된 프로세스의 개수)
- (3) 반환 시간(작업의 제출부터 완료되기까지의 소모 시간)
- (4) 대기 시간(프로세스가 준비 큐에서 스케줄링이 될 때까지 기다리는 방식)
- (5) 응답 시간(대화형 시스템에서 터미널을 통해 입력한 명령의 처리 결과가 나올 때까지의 시간)

[2] CPU 스케줄링 기법

(1) FCFS

- 프로세스들이 준비 큐에 도착한 순서대로 CPU를 할당하는 비선점 방식
- 공정하지만 짧은 작업이 긴 작업을, 중요한 작업이 중요하지 않은 작업을 기다리는 단점이 발생



(2) Round-Robin(라운드 로빈)

- FCFS 기법으로 처리하되 프로세스마다 동일한 CPU시간을 할당하는 선점 방식
- 프로세스가 할당된 시간 내에 처리를 완료하지 못한 경우 대기 중인 프로세스에게 CPU 양도 후 준비 큐의 맨 뒤로 이동함
- 타이머 인터럽트를 처리하는 오버헤드를 감수해야 하나 응답 시간이 짧아지는 장점이 존재
- 실시간 시스템이나 대화식 시분할 시스템에 유리

(3) SJF(Shortest Job First)

- 준비 큐의 대기 프로세스 중 예상 실행 시간이 가장 짧은 것을 우선 처리하는 비선점 방식
- 작업들의 평균 대기 시간이 최소가 되는 최적의 스케줄링 기법
- 긴 작업의 대기 시간이 장기화될 가능성이 크고 시간 예측이 어려움
- 대화식 시분할 시스템에는 부적합함

(4) SRT(Shortest remaining Time)

- 준비 큐 대기 프로세스 중 잔여 실행 시간이 가장 짧을 것으로 예상되는 프로세스를 우선 처리하는 선점 방식
- 처리 시간이 긴 작업은 SJF보다 대기 시간이 길어지고 기아 현상 발생 가능성이 존재함

(5) HRN(Highest Response ration Next)

- SJF 기법의 단점을 보완한 비선점 방식
- 스케줄링 시점에서 매번 프로세스의 응답률을 계산한 후 가장 높은 응답률을 가진 프로세스를 우선 처리함(응답률 = (대기시간+서비스시간)/서비스 시간)
- 입출력 프로세스에 유리함



(6) 우선순위

- 프로세스마다 우선순위를 부여하고 최상위 우선순위 프로세스를 먼저 처리하는 비선점 방식
- 동일한 우선순위를 가진 프로세스들은 FCFS 방식으로 CPU를 할당 받음
- 시간제한, 메모리 요구, 프로세스의 중요성들을 우선순위로 결정
- 무한 정지, 기아 상태 발생 가능성 존재. 기아상태는 에이징 기법으로 해결 가능



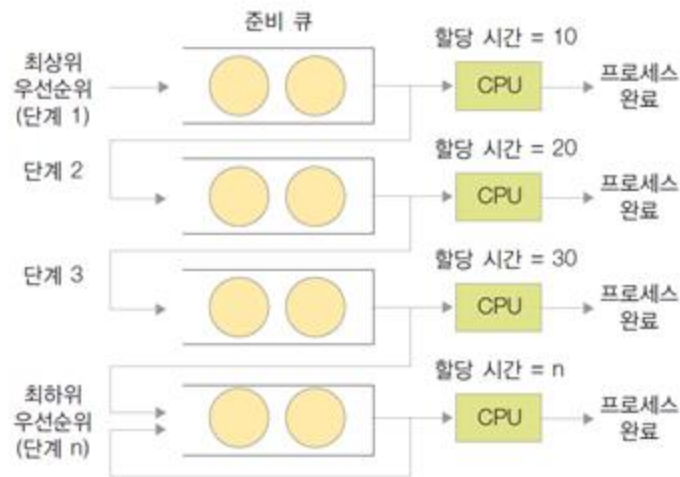
(7) MLQ(Multi Level Queue)

- 작업들을 여러 종류의 등급으로 분류하여 단계별 큐를 이용하는 기법으로 선점 방식
- 프로세스의 우선순위에 따라 시스템 프로세스, 대화형 프로세스, 학생 프로세스로 나누어 준비상태큐를 분류하며 각 큐마다 별도의 스케줄링 알고리즘을 적용함



(8) MFQ(Multi-level Feedback Queue)

- 입출력 위주 프로세스, CPU 위주 프로세스 특성에 따라 CPU 할당량을 다르게 부여하는 선점방식
- 새로운 프로세스는 최고 우선순위를 가지는 단계1에서 수행을 시작하고 CPU 할당량 내에 작업을 완료하지 못하면 단계2로 이동
- 단계 n에서 작업이 완료될 때까지 라운드 로빈 방식으로 반복 처리
- 작업들은 상위 큐에서 하위 큐로 에이징 기법으로 상위 큐로 이동할 수 있음
- 짧은 작업에 우선권을 부여함



<3> 기억장치 관리

[1] 주기억장치 관리법

(1) 단일 프로그래밍 기법

- 운영 체제와 하나의 사용자 프로그램만 주기억 장치에 적재됨
- 초기 컴퓨터 시스템에서 사용하던 가장 단순한 방식
- 주기억 장치의 빈 공간이 있어도 다른 사용자 프로그램을 실행할 수 없어 자원낭비가 심함
- 사용자 프로그램의 크기가 주기억장치의 용량을 초과할 수 없으므로, 주기억장치의 용량보다 작은 프로그램만 실행 가능
- 단일 프로그래밍 기법에는 오버레이와 스와핑이 있음

(2) 다중 프로그래밍 기법

1) 고정 분할 다중 프로그래밍 기법

- 주기억장치를 다수의 고정된 크기로 나눠서 실행 중인 여러 프로세스에게 할당함
- 작업과 분할의 크기가 일치하지 않아 사용되지 않는 빈 공간이 생기거나 분할이 너무 작아서 작업을 적재하지 못하는 단편화 현상이 발생함
- 단편화 현상은 분할에 작업을 적재한 후 빈 공간이 남는 내부 단편화와 적재할 작업보다 분할의 크기가 작아서 분할이 빈 공간으로 남게 되는 외부 단편화로 구분 할 수 있음

2) 가변 분할 다중 프로그래밍 기법

- 고정된 분할의 경계를 없애고 각 작업에게 필요한 만큼의 기억 공간을 할당함
- 작업이 완료되면 사용되지 않는 기억 공간을 회수하여 관리함
- 초반에는 기억 공간의 낭비가 없지만 새로운 작업 적재, 기억 공간 회수의 반복에 따라 단편화 현상이 발생하며 이를 방지하기 위해 통합과 집약 기법을 사용함

[2] 주기억장치 관리 전략

(1) 반입 전략

- 보조 기억장치의 프로그램이나 데이터를 언제 주기억장치에 적재할 것인지를 결정하는 기법

1) 요구 반입

- 현재 실행되는 프로그램에 의해 참조될 때 프로그램이나 데이터를 주기억장치에 적재
- 프로그램이나 데이터들의 요구가 있을 때만 주기억장치에 적재
- 오버헤드는 적으나 페이지 할당을 위한 대기시간 소요가 큼

2) 예상 반입

- 현재 실행되는 프로그램에 의해 참조될 가능성이 큰 프로그램이나 데이터,
미래에 요구할 가능성이 높은 프로그램이나 데이터를 예상하여 주기억장치에 적재
- 예측 결정이 맞으면 프로세스 실행 시간이 크게 감소, 예측 결정이 맞지 않으면 오버헤드가 큼

(2) 배치 전략

- 새로 가져온 프로그램이나 데이터의 주기억장치의 배치 장소를 결정하는 기법

1) 최초 적합

- 새로 반입된 프로그램이나 데이터를 주기억장치의 가용공간들 중에서 첫 번째 빈 공간에 배치
- 빈 공간을 찾기 위해 기억 공간 전체를 탐색할 필요가 없어 배치 결정이 빠름

2) 최적 적합

- 새로 반입된 프로그램이나 데이터를 주기억장치의 가용공간들 중에서 가장 작은 공간에 배치
- 기억 공간을 반만 탐색해도 적합한 공간을 찾을 수 있어 기억 공간 낭비를 최소화할 수 있음
- 빈 공간이 크기 순으로 정렬되어 있지 않으면 빈 공간 리스트를 모두 검색해야 하기 때문에,
빈 공간을 항상 크기 순으로 정렬해야 함

3) 최악 적합

- 새로 반입된 프로그램이나 데이터를 주기억장치의 가용공간들 중에서 가장 큰 공간에 배치
- 배치 후 남은 공간이 여전히 크기 때문에 다른 프로그램이나 데이터의 적재에 활용할 수 있음
- 커다란 프로그램을 적재할 공간이 없어지고, 빈 공간을 항상 크기가 작아지는 순으로 정렬

(3) 교체 전략

- 주기억장치에 새로 반입될 프로그램이나 데이터를 배치할 빈 공간이 없을 때 배치 공간을 마련하기 위해 어떤 프로그램이나 데이터를 주기억장치에서 제거할 것인지 결정하는 기법
- OPT, FIFO, LRU, NUR, LFU, SCR 등이 있음

[3] 가상기억장치 구현 기법

- 가상기억장치는 프로그램 전체가 주기억장치에 존재하지 않아도 실행이 가능하게 만들기 위해,
현재 실행 중인 프로그램의 일부는 주기억장치에 적재하고, 나머지는 보조기억장치에 유지함
- 가상기억장치에서 프로세스가 참조하는 가상 주소는 주소 사상 함수에 의해 실행 중에
주기억장치에서 사용할 수 있는 실제 주소로 변환됨

(1) 페이징 기법

- 가상기억장치를 일정한 크기의 페이지로 나눠서 관리함
- 주기억장치는 페이지와 크기가 동일한 페이지 프레임으로 분할하여 사용하는데,
일반적인 페이지 크기는 1~4KB

(2) 세그먼테이션 기법

- 프로그램이나 데이터를 용도에 따라 가변적인 크기로 분할하여 관리하는 것
- 가변적인 크기로 분할된 프로그램이나 데이터 블록을 세그먼트라고 함

(3) 페이징/세그먼테이션 혼합 기법

- 모든 세그먼트를 페이지 단위로 다시 분할하여 외부 단편화 현상을 해결함
- 세그먼트의 크기는 페이지의 정수 배
- 외부 단편화 현상은 제거했으나 내부 단편화 현상은 여전히 존재함
- 연관기억장치 사상표가 차지하는 기억 공간의 오버헤드가 증가함