11장 CPU 스케줄링

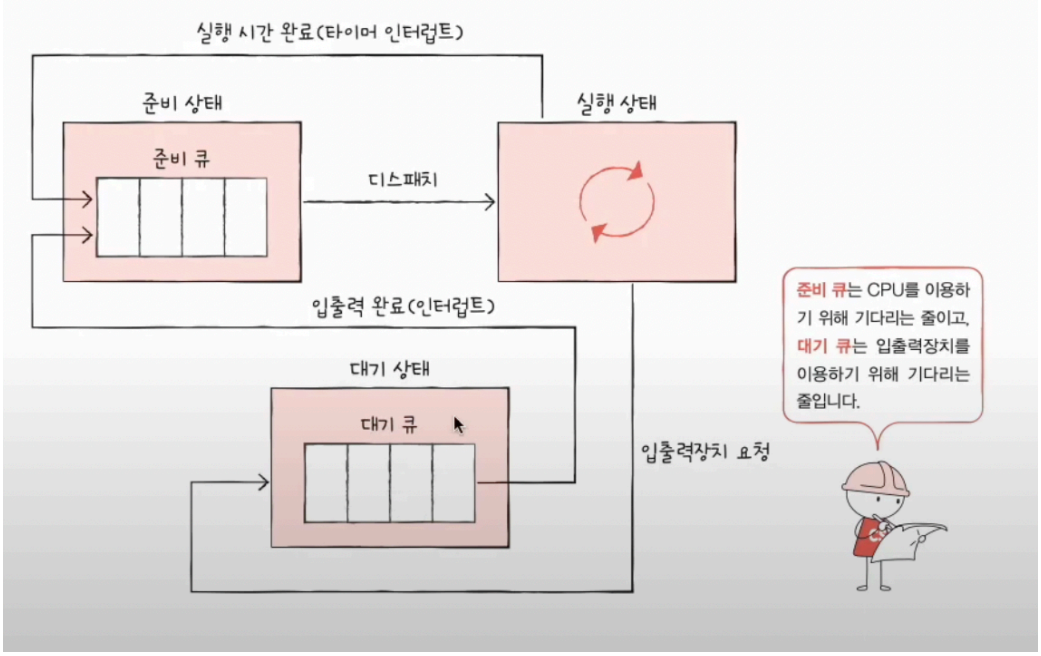
- 공정하고 합리적으로 CPU 자원을 배분하는 방법을 의미

프로세스 우선순위

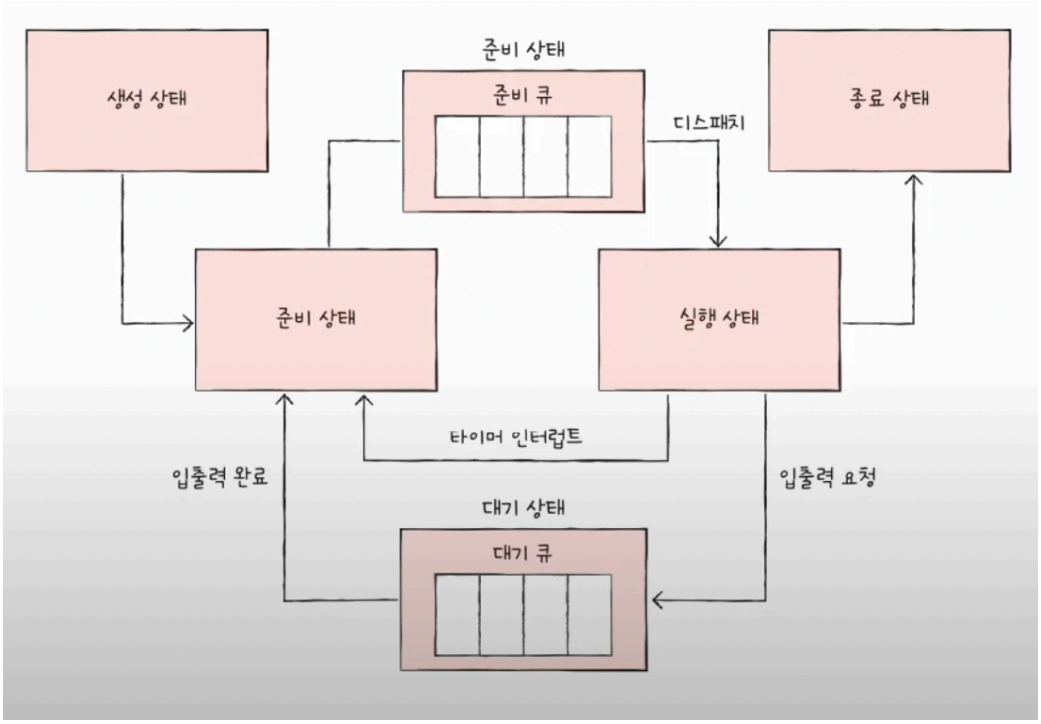
- 우선순위가 높은 프로세스 : 입출력 작업이 많은 프로세스
- 입출력 집중 프로세스
    - 입출력장치를 사용하는 작업 : 입출력 버스트
- CPU 집중 프로세스
    - CPU를 이용하는 작업 : CPU 버스트
- CPU 집중 프로세스와 입출력 집중 프로세스가 동시에 CPU 자원을 요구한 경우
    - 입출력 집중 프로세스를 가능한 한 빨리 실행시켜 입출력장치를 끊임없이 작동시키고, 그 다음 CPU 집중 프로세스에 CPU를 할당하는 것이 더 효율적임
    - 입출력장치가 입출력 작업을 완료하기 전까지는 입출력 집중 프로세스는 대기 상태가 될 예정이기 때문에 입출력 집중 프로세스를 먼저 처리해버리면 다른 프로세스가 CPU를 사용할 수 있기 떄문
- 운영체제는 프로세스마다 우선순위를 부여함

스케줄링 큐

- 운영체제는 메모리에 적재되고 싶은 프로세스들을 큐에 삽입하여 줄을 세우고, CPU를 이용하고 싶은 프로세스들 또한 큐에 삽입하여 줄을 세우고, 특정 입출력장치를 이용하고 싶은 프로세스들 역시 큐에 삽입하여 줄을 세움



- 준비 큐 : CPU를 이용하고 싶은 프로세스들이 서는 줄
- 대기 큐 : 입출력장치를 이용하기 위해 대기 상태에 접어든 프로세스들이 서는 줄



선점형과 비선점형 스케줄링

- **선점형 프로세스** : 프로세스가 CPU를 비롯한 자원을 사용하고 있더라도 운영체제가 프로세스로부터 자원을 강제로 빼앗아 다른 프로세스에 할당할 수 있는 스케줄링 방식
    - 어느 하나의 프로세스가 자원 사용을 독점할 수 없는 스케줄링 방식

    - **장점** : 어느 한 프로세스의 자원 독점을 막고 프로세스들에 골고루 자원을 배분할 수 있음
    - **단점** : 문맥 교환 과정에서 오버헤드가 발생할 수 있음
- **비선점형 스케줄링** : 하나의 프로세스가 자원을 사용하고 있다면 그 프로세스가 종료되거나 스스로 대기 상태에 접어들기 전까진 다른 프로세스가 끼어들 수 없는 스케줄링 방식을 의미
    - 하나의 프로세스가 자원 사용을 독점할 수 있는 스케줄링 방식
    - **장점** : 문맥 교환의 횟수가 선점형 스케줄링보다 작기 때문에 문맥 교환에서 발생하는 오버헤드는 선점형 스케줄링보다 적음
    - **단점** : 하나의 프로세스가 자원을 사용 중이라면 당장 자원을 사용해야 하는 상황에서도 무작정 기다리는 수밖에 없음

CPU 스케줄링 알고리즘

선입선출 스케줄링 (FCFC 스케줄링)

- 준비 큐에 삽입된 순서대로 프로세스들을 처리하는 비선점형 스케줄링 방식
- 프로세스들이 기다리는 시간이 매우 길어질 수 있다는 점에서 부작용이 있는 방식
- 호위 효과 : CPU를 오래 사용하는 프로세스가 먼저 도착하면 다른 프로세드는 그 프로세스가 CPU를 사용하는 동안 무작정 기다릴 수 밖에 없음

최단 작업 우선 스케줄링 (SJF 스케줄링)

- CPU 사용 시간이 긴 프로세스는 나중에 실행하고, CPU 사용 시간이 짧은 간단한 프로세스 먼저 실행

라운드 로빈 스케줄링

- 선입 선처리 스케줄링 + 타임 슬라이스
- 타임 슬라이스 : 각 프로세스가 CPU를 사용할 수 있는 정해진 시간을 의미
- 정해진 타임 슬라이스만큼의 시간 동안 돌아가면 CPU를 이용하는 선점형 스케줄링
- 정해진 시간을 모두 사용하였음에도 아직 프로세스가 완료되지 않았다면 다시 큐의 맨 뒤에 삽입됨 → 문맥 교환 발생
- 타임 스라이스의 크기가 매우 중요
    - 지나치게 크면 선입 선처리 스케줄링과 다를 바 없어 호위 효과가 생길 여지가 있고, 타임 슬라이스가 지나치게 작으면 문맥 교환에 발생하는 비용이 커 PCU는 프로세

스를 처리하는 일보다 프로세스를 전환하는 데에 온 힘을 다 쓰게 됨

최소 잔여 시간 우선 스케줄링(SRT 스케줄링)

- 최단 작업 우선 스케줄링 알고리즘 + 라운드 로빈 알고리즘
- 최소 잔여 시간 우선 스케줄링 하에서 프로세스들은 정해진 타임 슬라이스만큼 CPU를 사용하되, CPU를 사용할 다음 프로세스로는 남아있는 작업 시간이 가장 적은 프로세스가 선택

우선순위 스케줄링

- 프로세스들에 우선순위를 부여하고, 가장 높은 우선순위를 가질 프로세스로부터 실행하는 스케줄링 알고리즘
- 근본적인 문제 : 우선순위가 높은 프로세스를 우선하여 처리하는 방식이게에 우선순위가 낮은 프로세스는 (준비 큐에 먼저 삽입되었음에도 불구하고) 우선순위가 높은 프로세스들에 의해 실행이 계속해서 연기될 수 있음 : `기아 현상`
- `에이징 기법` : 오랫동안 대기한 프로세스의 우선순위를 점차 높이는 방식

다단계 큐 스케줄링

- 우선순위별로 준비 큐를 여러 개 사용하는 스케줄링 방식
- 우선순위가 가장 높은 큐에 있는 프로세스들을 먼저 처리하고, 우선순위가 가장 높은 큐에 비어 있으면 그다음 우선순위 큐에 있는 프로세스들을 처리함

다단계 피드백 큐 스케줄링

- 다단계 큐 스케줄링에서는 프로세스들이 큐 사이를 이동할 수 없음 → 우선순위가 낮은 프로세스는 계속 연기될 여지가 있음
- 새로 준비 상태가 된 프로세스가 있다면 우선 우선순위가 가장 높은 우선순위 큐에 삽입되고 일정 시간 동안 실행됨
- 프로세스가 해당 큐에서 실행이 끝나지 않는다면 다음 우선순위 큐에 삽입되어 실행됨.
- 즉, CPU를 오래 사용해야 하는 프로세스는 점차 우선순위가 낮아짐. CPU를 비교적 적게 사용하는 입출력 집중 프로세스들은 자연스레 우선순위가 높은 큐에서 실행이 끝남