**Chapter 11 CPU 스케줄링**

***11-1*** **CPU 스케줄링 개요**

**CPU 스케줄링**: 운영체제가 프로세스들에게 공정하고 합리적으로 CPU 자원을 배분하는 것

**<프로세스 우선순위>**

**우선순위**가 **높은** 프로세스란 빨리 처리해야 하는 프로세스를 의미

ex) 입출력 작업이 많은 프로세스

**입출력 집중 프로세스**: 입출력 작업이 많은 프로세스. 실행 상태보다는 입출력을 위한 대기 상태에 더 많이 머무르게 됨

ex) 비디오 재생, 디스크 백업

**CPU 집중 프로세스**: CPU 작업이 많은 프로세스. 대기 상태보다는 실행 상태에 더 많이 머무르게 됨

ex) 복잡한 수학 연산, 컴파일, 그래픽 처리 작업

**각각의 집중 프로세스가 동시에 CPU 자원을 요구했다면?** 입출력 집중 프로세스를 가능한 한 빨리 실행시켜 입출력장치를 끊임없이 작동시키고, 그 다음 CPU 집중 프로세스에 집중적으로 CPU를 할당하는 것이 더 효율적.

.

상황과 프로세스의 중요도에 맞게 프로세스가 CPU를 이용할 수 있도록 하기 위해 운영체제는 프로세스마다 **우선순위**를 부여 -> 각 프로세스의 **PCB**에 명시. 우선순위**(Process Priority)** 기준으로 먼저 처리할 프로세스 결정

**<스케줄링 큐>**

프로세스의 우선순위를 찾기 위해 운영체제가 일일이 모든 프로세스의 PCB를 뒤적거리는 것은 비효율적 -> CPU사용을 원하는 프로세스, 메모리 적재를 원하는 프로세스, 입출력장치 사용을 원하는 프로세스를 모두 **줄 세움** -> 이 줄을 **스케줄링 큐**로 구현하고 관리

**준비 큐**: CPU를 이용하고 싶은 프로세스들이 서는 줄

**대기 큐**: 입출력장치를 이용하기 위해 대기 상태에 접어든 프로세스들이 서는 줄

**<선점형과 비선점형 스케줄링>**

**선점형 스케줄링**:

* 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 없음
* 현재 대부분의 운영체제는 선점형 스케줄링 방식
* 골고루 자원을 배분할 수 있다는 장점
* 문맥 교환 과정에서 오버헤드가 발생할 수 있는 단점

**비선점형 스케줄링**:

* 하나의 프로세스가 자원을 사용하고 있다면 그 프로세스가 종료되거나 스스로 대기 상태에 접어들기 전까진 다른 프로세스가 끼어들 수 없는 스케줄링 방식
* 하나의 프로세스가 자원 사용을 독점할 수 있음
* 문맥 교환 시 발생하는 오버헤드는 선점형 스케줄링보다 적다는 장점
* 모든 프로세스가 골고루 자원을 사용할 수 없다는 단점

***11-2*** **CPU 스케줄링 알고리즘**

**<스케줄링 알고리즘의 종류>**

1. **선입 선처리 스케줄링(FCFS)**

* 준비 큐에 삽입된 순서대로 프로세스들을 처리하는 비선점형 스케줄링 방식
* CPU를 먼저 요청한 프로세스부터 CPU를 할당
* 때때로 프로세스들이 기다리는 시간이 매우 길어질 수 있음
* **호위 효과**가 있을 수 있음

**호위 효과**: 우선 순위가 높은 프로세스의 긴 처리시간으로 우선 순위가 낮은 프로세스들이 오래 기다리게 되는 현상

ex) A-B-C-D 순으로 프로세스가 실행될때 각각 16초,5초,2초,7초가 걸린다면 프로세스 C는 2초를 처리하기 위해 21초를 기다려야 함

1. **최단 작업 우선 스케줄링(SJF)**

* 호위 효과 방지
* CPU의 이용 시간의 길이가 가장 짧은 프로세스부터 실행하는 스케줄링 방식
* 비선점형 스케줄링 알고리즘이지만 선점형으로 구현될 수도 있음(선점형 최단 작업 우선 스케줄링)

ex) 프로세스 A,B,C,D가 실행될때 각각 16초,5초,2초,7초가 걸린다면 C-B-D-A 순으로 처리

1. **라운드 로빈 스케줄링**

* 선입 선처리 스케줄링에 **타임 슬라이스**라는 개념이 더해진 스케줄링 방식

**타임 슬라이스**: 각 프로세스가 CPU를 사용할 수 있는 정해진 시간

* 정해진 타임 슬라이스만큼의 시간 동안 돌아가며 CPU를 이용하는 선점형 스케줄링 방식
* 정해진 시간을 모두 사용하였음에도 아직 프로세스가 완료되지 않았다면? 다시 큐의 맨 뒤에 삽입 -> 문맥 교환 발생

ex) 프로세스 A,B,C,D가 실행될때 각각 16초,5초,2초,7초가 걸리고 타임 슬라이스가 5초라면 ->

1회전에서 A가 11초 잔여, B 완료, C완료, D가 2초 잔여

2회전에서 A는 6초 잔여, D 완료

3회전에서 A는 1초 잔여

4회전에서 A는 완료

1. **최소 잔여 시간 우선 스케줄링(SRT)**

* 최단 작업 우선 스케줄링 알고리즘과 라운드 로빈 알고리즘을 합친 스케줄링 방식
* 최소 잔여 시간 우선 스케줄링 하에서 프로세스들은 정해진 타임 슬라이스만큼 CPU를 사용하되, CPU를 사용할 다음 프로세스로는 남아있는 작업 시간이 가장 적은 프로세스가 선택

ex) 프로세스 A,B,C,D가 실행될때 각각 16초,5초,2초,7초가 걸리고 타임 슬라이스가 5초라면 ->

1회전에서 A가 11초 잔여, B 완료, C완료, D가 2초 잔여

2회전에서 남아있는 시간이 적은 순서대로 D먼저 실행 후 D완료 다음 A프로세스 6초 잔여

3회전에서 A는 1초 잔여

4회전에서 A는 완료

1. **우선순위 스케줄링**

* 프로세스들에 우선순위를 부여하고, 가장 높은 우선순위를 가진 프로세스부터 실행하는 스케줄링 알고리즘
* **기아현상**: 우선순위가 낮은 프로세스는 우선순위가 높은 프로세스들에 의해 실행이 계속해서 연기 되는 현상:
* **에이징**: 기아현상을 방지하기 위한 기법. 오랫동안 대기한 프로세스의 우선순위를 점차 높이는 방식

1. **다단계 큐 스케줄링**

* 우선순위별로 준비 큐를 여러 개 사용하는 스케줄링 방식
* 우선순위가 가장 높은 큐에 있는 프로세스들을 먼저 처리. 우선순위가 가장 높은 큐가 비어 있으면 그다음 우선순위 큐에 있는 프로세스를 처리
* 큐별로 타임 슬라이스를 여러 개 지정 가능, 큐마다 다른 스케줄링 알고리즘을 사용할 수도 있음

1. **다단계 피드백 큐 스케줄링**

* 다단계 큐 스케줄링의 기아현상을 방지하기 위해 발전된 형태
* 프로세스들이 큐 사이를 이동할 수 있음
* 새로 준비 상태가 된 프로세스가 있다면 우선 우선순위가 가장 높은 우선순위 큐에 삽입되고 일정 시간동안 실행 -> 만약 프로세스가 해당 큐에서 실행이 끝나지 않는다면 다음 우선순위 큐에 삽입되어 실행
* CPU 집중 프로세스들은 자연스레 우선순위가 낮아지고 입출력 집중 프로세스들은 우선순위가 높은 큐에서 실행
* 구현이 복잡하지만, 가장 일반적인 CPU 스케줄링 알고리즘

