

## 2 운영체제\_CPU 스케줄링

- CPU가 유휴 상태가 될때마다, 운영체제는 준비 큐에 있는 프로세스 중 하나를 선택해 실행 시킴.
- 이때 CPU의 유휴 시간을 최소화하고 시스템 효율을 높이기 위한 전략을 세우는데, 이 전략이 바로 CPU 스케줄링 알고리즘임.
- 프로세스를 관리하는 큐 종류
  1. 작업 큐(Job Queue)
    - 시스템에 제출된 모든 프로세스가 저장되는 큐
    - 디스크 등 보조기억장치에 존재하며, 프로세스의 생성 순서대로 저장
    - 장기 스케줄러가 관리
  2. 준비 큐(Ready Queue)
    - 메모리에 적재되어 CPU 할당을 기다리는 프로세스들이 저장되는 큐
    - 단기 스케줄러가 관리 하며, 다양한 스케줄링 알고리즘에 따라 실행 순서 결정
  3. 대기 큐(Wait Queue)
    - I/O 작업이나 특정 이벤트 발생을 기다리는 프로세스들이 저장되는 큐
    - 각 I/O 장치마다 대기 큐가 존재
    - 이벤트가 발생하면 해당 프로세스는 준비 큐로 이동
- 스케줄러의 종류
  - 프로세스 관리의 시간적 범위와 실행 빈도에 따라 장기, 중기,단기 스케줄러로 나뉨
  - 실행 빈도는 장기 → 단기 순으로 높음
  1. 장기 스케줄러
    - 디스크에 있는 작업을 어떤 순서로 메모리에 가져올지 결정
    - 작업 큐 → 준비 큐로 프로세스를 이동시키는 역할
    - 디스크와 메모리 간 스케줄링 담당
    - 프로세스 상태 변화: 생성 → 준비

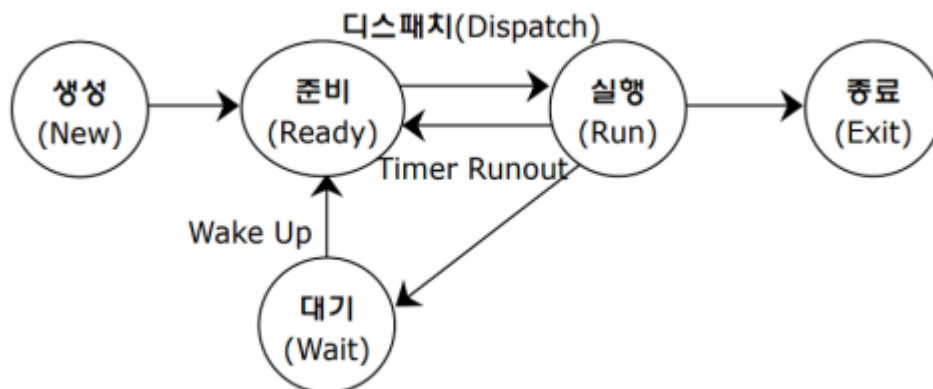
- 현대의 시분할 시스템에서는 거의 사용되지 않으며, 배치 처리 시스템에 제한적으로 사용

## 2. 중기 스케줄러

- 메모리에 적재된 프로세스 수를 조절해 시스템 성능을 최적화
- 스왑핑 기법을 사용하여 프로세스를 메모리 ↔ 디스크로 이동
- 프로세스 상태 변화: 준비 ↔ 중단

## 3. 단기 스케줄러

- 준비 큐에 있는 프로세스 중 CPU를 할당할 대상을 결정
- 시분할 시스템에서 주로 사용되며, 실행 빈도가 가장 높음
- 메모리와 CPU 사이의 스케줄링 담당
- 프로세스 상태 변화: 준비 → 실행 → 대기 → 준비



### • 시스템 동작 흐름 요약

1. 새로운 프로세스가 작업 큐에 저장
2. 장기 스케줄러가 작업 큐에서 프로세스를 선택 해 준비 큐로 이동
3. 단기 스케줄러가 준비 큐에서 CPU에 할당할 프로세스 선택
4. 디스패처가 선택된 프로세스에 CPU를 할당하고 문맥 교환 수행
5. 실행 중 I/O 작업이 필요하면 대기 큐로 이동
6. I/O 완료 시 다시 준비 큐로 복귀

- 프로세스 상태
  - New: 새로 생성된 상태, 곧 준비 큐로 이동 예정
  - Ready: CPU 할당을 기다리는 상태
  - Run: CPU를 할당받아 실행 중인 상태
  - Wait: 실행 중 I/O 작업 등으로 대기 중인 상태
  - Exit: 실행이 종료된 상태
  
- 스케줄링의 구분
  - CPU를 프로세스에 할당하는 방식에 따라 선점과 비선점으로 나눔
  
- 선점형 스케줄링(Preemptive)
  - 실행 중인 프로세스가 끝나기 전에 다른 프로세스가 CPU를 강제로 점유 가능
  - 우선순위가 높거나 긴급한 처리를 요할 때 사용
  - 장점
    - 응답 시간이 중요한 시스템에 적합
    - 우선순위가 높은 프로세스를 신속히 처리가능
  - 단점
    - 잦은 문맥 교환(Context Switching)으로 인한 오버헤드 발생
    - 우선순위 관리가 복잡해질 수 있음
  - 예시
    - 시분할 시스템, 실시간 시스템
    - RR, SRTF
  
- 비선점형 스케줄링(Non-Preemptive)
  - 실행 중인 프로세스가 종료될 때까지 CPU를 반환하지 않음
  - 다른 프로세스는 CPU를 자발적으로 반환할 때까지 대기
  - 장점
    - 문맥 교환 오버헤드가 적음

- 관리가 단순함
- 단점
  - 응답 시간이 느림
  - 긴 작업이 먼저 수행되면 짧은 작업이 오랫동안 대기
- 예시
  - 배치 처리 시스템
  - FCFS, SJF
- 기아 상태
  - 우선순위가 낮은 프로세스가 계속해서 CPU를 할당받지 못하는 상태
  - 우선순위 기반 스케줄링에서 자주 발생
  - 해결방법: 에이징 기법
    - 오랫동안 대기한 프로세스의 우선순위를 점차 높여주는 방식
    - 대기 시간이 길수록 우선순위를 높여 공정성을 보장
- 주요 스케줄링 알고리즘
  1. FCFS (First Come, First Served)
    - 준비 큐에 도착한 순서대로 CPU를 할당(FIFO 방식)
    - 비선점형
    - 단순하고 공정
    - 긴 작업이 오면 다른 프로세스의 대기시간이 길어짐
    - 배치 처리 시스템
  2. SJF(Shortest Job First)
    - CPU 사용 시간이 가장 짧은 프로세스에 우선 할당
    - 평균 대기 시간을 최소화
    - 비선점형
    - CPU 점유 시간을 예측해야 함(과거 데이터 기반)
    - CPU 사용 시간이 긴 프로세스는 대기 시간이 길어질 수 있음

- 평균 대기 시간 최소화
- 긴 작업의 기아 상태 발생 가능
- 배치 시스템

### 3. SRTF(Shortest Remaining Time First)

- SJF의 선점형 버전
- 실행 중인 프로세스보다 새로 도착한 프로세스의 남은 실행 시간이 짧으면 CPU를 선점
- 평균 대기 시간 최소화
- 잦은 문맥 교환, 긴 작업의 기아 가능성
- 실시간 시스템

### 4. 라운드 로빈(Round Robin)

- 도착 순서대로 일정한 시간 할당량 만큼 CPU 사용
- 할당 시간이 끝나면 CPU를 다른 프로세스에 넘김
- 시분할 시스템에 적합, 선점형 방식
- FCFS에 시간 할당량 개념을 추가한 방식
- 시간 할당량이 너무 크면 FCFS와 유사
- 너무 작으면 문맥 교환 오버헤드 증가
- 공정성, 응답 속도 우수
- 타임 슬라이스 설정이 어렵고 오버헤드 발생 가능
- 시분할 및 실시간 시스템
  - 스케줄링 계산 방식
    - CPU 스케줄링 참고
- 멀티 레벨 큐와 멀티 레벨 피드백 큐 스케줄링
  - 차주 스터디 주제로 선정