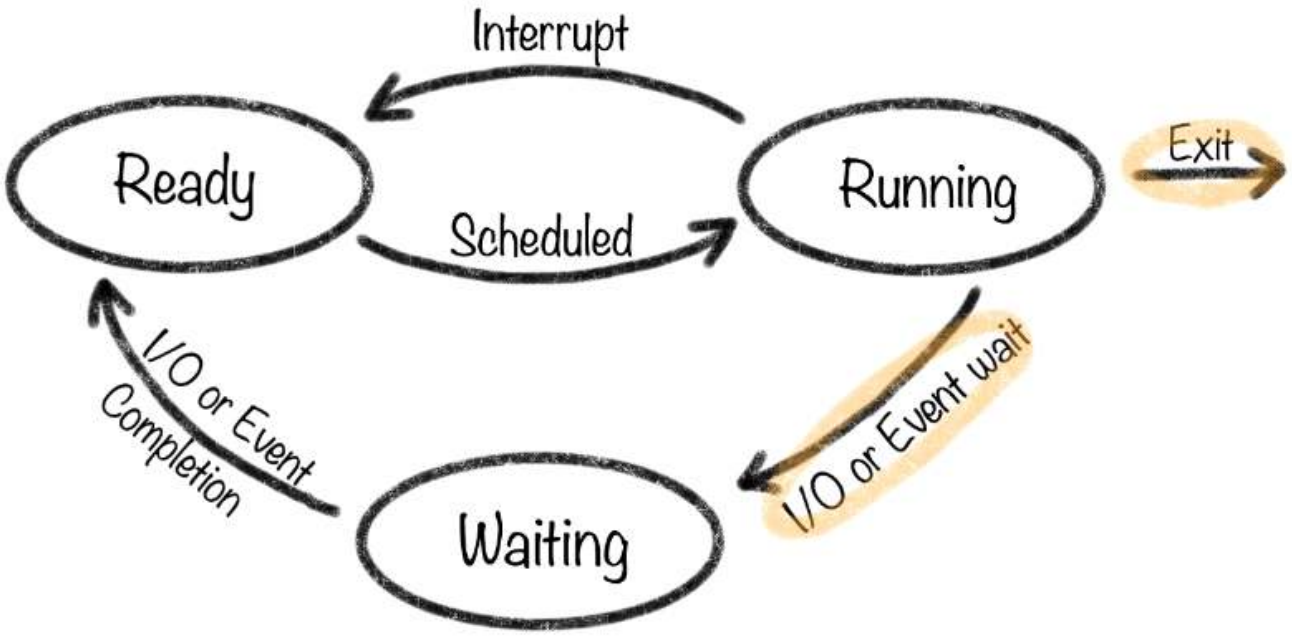
나올 수 있는 면접 질문

Q. 선점형 스케줄링과 비선점형 스케줄링의 차이가 무엇인가요?

선점형 스케줄링이란 실행중인 프로세스나 스레드를 강제로 중단시키고 다른 프로세스를 실행시키는 것이다.
비선점형 스케줄링은 그렇지 않은 것이다.

현대 운영체제는 선점형 스케줄링을 사용한다.
비선점형 스케줄링은 컨텍스트 스위칭이 적어 오버헤드가 적다는 장점이 있지만 간단한 작업에서 오래 기다릴 수도 있는 문제가 있기 때문이다.

비선점 스케줄링이란?



> 프로세스가 자원을 할당받았을 경우 자원을 스스로 반납할 때 까지 계속 그 자원을 사용하도록 허용하는 정책

작업 실행 시간 전체 또는 한 번의 CPU배당에 대해 적용되며, 비선점 스케줄링의 예로는 FCFS, SJF, HRN, 우선순위 스케줄링 등이 있다.

FCFS 스케줄링 (First Come First Served Scheduling)

- 프로세스의 도착순으로 CPU를 배정
- Batch 작업등, 장기 스케줄러에서 사용 가능

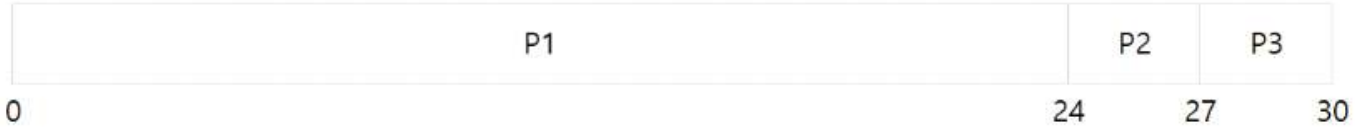
> Batch 작업
데이터를 실시간으로 처리하는게 아니라, 일괄적으로 모아서 한번에 처리하는 작업

> 장기 스케줄러
작업 스케줄러라고도 부르며, 어떤 프로세스를 준비 큐에 삽입할지를 결정하는 역할을 함. 디스크에서 하나의 프로그램을 가져와 커널에 등록하면 프로세스가 되는데 이때 디스크에서 어떤 프로그램을 가져와 커널에 등록할지(준비큐에 등록할지) 결정

- 수행 중인 긴 작업을 여러 개의 짧은 작업들이 기다리게 되는 호위효과 (Convoy Effect)의 문제가 발생할 수 있음

작업	Burst Time
P1	24
P2	3
P3	3

위와 같은 작업의 도착 순서는 P1 -> P2 -> P3 이고, 간트차트는 다음과 같음.



- 평균 반환시간 = (24+27+30) / 3 = 27
- 평균 대기시간 = (0+24+27) / 3 = 17

최단 작업 우선 스케줄링 (SJF, Shortest Job First Scheduling)

- 대기하는 작업 중 CPU Burst Time이 가장 작은 작업에 CPU를 할당하는 기법
- 평균 대기 시간에 있어 최적의 알고리즘

작업	Burst TIme
P1	6
P2	3
P3	8
P4	7

위와 같은 작업이 있다면 순서는 Burst Time이 작은 P2 -> P1 -> P4 -> P3 이다. 간트차트로 나타낸다면 다음과 같다.

- 평균 반환시간 = (3+9+16+24) / 4 = 13
- 평균 대기시간 = (3+0+16+9) / 4 = 7

HRN 스케줄링

- 짧은 작업에 유리한 SJF의 단점을 개선 한 기법, 각 작업의 우선순위로 서비스 해주는 스케줄링
- SJF의 약점을 보완한 기법으로 긴 작업과 짧은 작업 간 불평등 완화
- 우선순위 계산 공식 = (대기시간 + 실행시간) / 실행시간
- 우선순위 계산 결과값이 높은 것 부터 우선 순위가 부여, 대기 시간이 긴 프로세스일 경우 계산 결과값이 높게 나옴

작업	대기시간	실행시간
P1	5	5
P2	10	6
P3	15	7
P4	20	8

P1 우선순위 = (5+5)/5 = 2
P2 우선순위 = (10+6)/6 = 2.66..
P3 우선순위 = (15+7)/7 = 3.14..
P4 우선순위 = (20+8)/8 = 3.5

작업 순서는 P4 -> P3 -> P2 -> P1

우선순위 스케줄링 (Priority Scheduling)

- 각 프로세스의 우선순위가 정해지면, 우선순위가 제일 높은 프로세스에게 CPU를 할당하되, 우선순위가 같은 경우 FCFS 방식을 적용한다.
- 일반적인 연산 위주 프로세스보다 입출력 위주 프로세스에게 높은 우선순위를 부여하여 대화성을 증진시킨

다.
- 우선순위가 높은 작업이 계속적으로 들어올 경우 우선순위가 낮은 작업은 준비상태에서 보장 없이 머물게 된다-기아상태. 이런 문제는 시스템에 머무는 시간이 증가할수록 우선순위를 높여주는 에이징으로 해결한다.

> 기아현상(STARVATION)이란?
자신보다 우선순위 높은 프로세스 때문에 오랫동안 CPU 할당을 받지 못하고 무한정 기다리게 되는 현상

> 에이징이란?
프로세스 자원을 기다리고 있는 시간에 비례하여 우선순위를 부여함으로써 무기한 연기의 문제를 방지하는 기법

- 우선순위 스케줄링 알고리즘은 일반적으로 가장 많이 쓰이는 스케줄링 기법이다.
- 내부적 우선순위 고려 : 제한 시간, 메모리 요구량, 사용하는 파일 수, 평균 CPU Burst에 대한 평균 I/O Burst의 비율 등
- 외부적 우선순위 고려 : 사용료, 정책적인 변수 등

작업	Burst Time	우선순위
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

위와 같은 작업이 있다면 순서는 우선 순위 순대로 가되, 우선순위가 같은 P1,P3의 경우 먼저 도착한 순인 P1 -> P3 순.

작업순서 : P2 -> P5 -> P1 -> P3 -> P4

- 평균 반환시간 = (16+1+18+19+6) / 5 = 12
- 평균 대기시간 = (6+0+16+18+1) / 5 = 8.2

선점 스케줄링이란?

- 선점 스케줄링이란 우선순위가 높은 프로세스가 현재 프로세스를 중지시키고 자신이 CPU를 점유하는 스케줄링 기법이다.
- 비교적 응답이 빠르다는 장점이 있으나, 처리 시간을 예측하기 힘들고 높은 우선순위 프로세스들이 계속 들어오는 경우 오버헤드가 발생하게 된다.
- 선점 스케줄링의 예로는 RR, SRT, 다단계 큐, 다단계 피드백 큐가 있음.

RR (Round Robin)

Round Robin Example:

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is 1 unit.

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2

0 10

P1 waiting time : 4

The average waiting time(AWT) : (4+6+6)/3=5.33

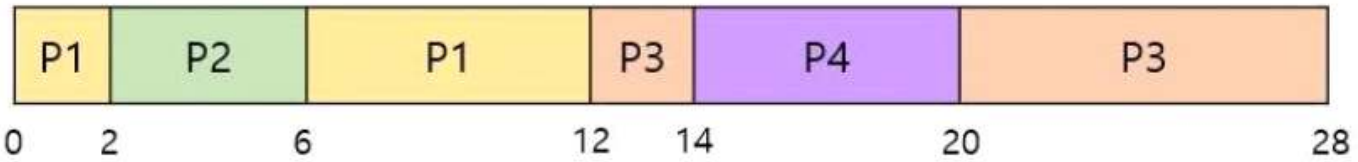
P2 waiting time: 6

P3 waiting time: 6

- 라운드 로빈은 프로세스마다 같은 크기의 CPU 시간을 할당받는다.
- 프로세스가 할당된 시간 내에 작업을 처리하지 못하면 준비 큐 리스트의 맨 뒤로 보내지며, CPU는 대기 중인 다음 프로세스로 넘어간다.
- 모든 프로세스가 CPU를 균등하게 점유하므로 기아현상이 발생하지 않는다.
- 할당된 시간이 크면 FCFS와 유사해지며, 작으면 잦은 문맥교환으로 인한 오버헤드가 발생한다.

SRT (Shortest Remaining Time)

✓ SRT Scheduling



Process	Arrival Time	Burst Time	Waiting Time	Turnaround Time
P1	0	8	0+4	12
P2	2	4	0	4
P3	10	10	2+6	18
P4	14	6	0	6

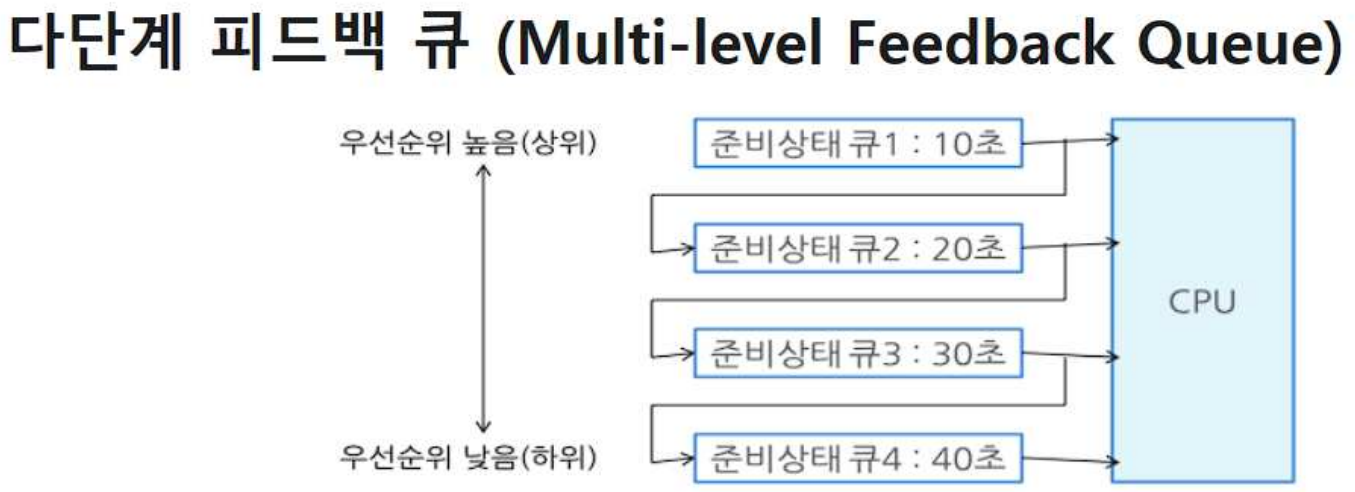
- Average Waiting Time : (4+0+8+0)/4 = 3
- Average Turnaround Time : (12+4+18+6)/4 = 10

- SRT는 위의 비선점형 스케줄링에서 언급한 SJF를 선점형으로 변경한 스케줄링 기법이다.
- 남은 시간이 가장 적은 프로세스를 먼저 수행한다.
  - 만약 더 짧은 처리 시간의 프로세스가 들어오면 실행 중인 프로세스를 중단시키고 더 짧은 처리시간의 프로세스를 처리한다.
- 처리 시간이 짧은 프로세스가 계속 추가되면 기아 현상이 발생할 수 있다.

다단계 큐 (Multi-level Queue)

- 작업들을 각 목적에 맞게 여러 종류의 그룹으로 분할을 한다.
- 각 그룹의 큐에는 독자적인 스케줄링 알고리즘(라운드 로빈, FCFS 등)을 사용할 수 있다.
- 더 높은 우선순위를 갖는 큐가 비워지지 않는 경우에는 기아 현상이 발생할 수 있다.

다단계 피드백 큐 (Multi-level Feedback Queue)



- 각 준비상태 큐마다 서로 다른 CPU 할당 시간을 부여하여 Time-out이 되면 처리하지 못한 프로세스를 다음 단계의 큐로 이동시킨다.
- 하위 단계로 내려갈수록 할당시간을 증가 시킨다.
- 마지막 단계에서는 라운드 로빈 또는 FCFS로 처리한다.
- 최하위 큐에서 너무 오래 대기 시 에이징 기법을 통해 상위로 이동시켜 처리한다.

결론

우리는 컴퓨터를 사용할 때 많은 프로세스들을 동시에 실행한다. 하지만 CPU는 한 번에 한 가지 작업만 처리하기에, 이 프로세스들을 동시에 처리하는 것처럼 보이기 위해 CPU는 어떤 프로세스에게 얼마나 많은 자원, 시간 할당을 해야할 지 결정해야 한다. 이를 결정하는 CPU 스케줄링 방식에 비선점, 선점 방식으로 분류하고 수많은 방식이 존재한다.

멀티스레드의 스케줄링은 운영체제가 처리하지 않기 때문에 프로그래머가 직접 동기화 문제에 대응해야한다.

그러나 CPU 스케줄링은 멀티스레드와는 다르게 운영체제가 사용자도 모르는 사이 자동으로 진행하는 작업이다.