8. CPU 스케줄링(scheduling)

CPU 스케줄링은 준비큐에 대기 중인 여러 프로세스들 중에서 어느 것을 CPU에 할당할 것인지를 다루는 것이다. CPU는 컴퓨터의 중요한 자원이므로 CPU 스케줄링은 운영체제 설계의 핵심 부분이기도 하다.

◈ 스케줄링 기법

선점 기법 (preemptive)	① 어떤 프로세스가 CPU를 점유하고 있을 때 다른 프로세스가 현재 프로세스를 중지시키고 자신이 CPU를 **강제로 점유**할 수 있다. ② 선점 기법은 높은 우선순위를 가진 프로세스들이 빠른 처리를 요구하는 시스템에 유용하다. 실시간, 시분할 시스템에 유용하다.**(대화식 시스템)** ③ 오버헤드가 많이 발생한다.
비선점 기법 (non_preemptive)	① 프로세스가 일단 CPU를 점유하게 되면 자신의 작업을 완료할 때까지 CPU를 사용한다. 즉, 다른 프로세스가 강제로 CPU를 빼앗을 수 없다. ② 프로세스의 우선순위가 없다.**(공정히 처리)** ③ 모든 프로세스를 비선점 기법으로 처리하면 응답시간을 쉽게 예측할 수 있다. ④ 짧은 작업(중요한 작업)이 긴 작업을 기다릴 수 있다. ⑤ 대화식 시스템에 적합하지 않다.

◈ 스케줄러(scheduler)

단기 스케줄러 (short term scheduler)	• CPU 스케줄러 또는 프로세스 스케줄러라고도 한다. • **준비**상태의 프로세스들 중에서 한 프로세스를 선택하여 **CPU를 할당**한다. • 프로세스들이 **빈번하게 교체**되므로 단기 스케줄러의 수행속도는 빨라야 한다.(해서, 단기 스케줄러)
장기 스케줄러 (long term scheduler)	• **작업** 스케줄러라고도 한다. • 디스크에 있는 모든 프로세스(작업)들의 상태를 파악하여, 다음에 수행할 프로세스를 선택하여 주기억장치로 적재시킨다. • 프로세스에게 필요한 시스템의 자원들을 분배한다. • **다중 프로그래밍 정도를 결정**한다. • 장기 스케줄러는 실행 빈도수가 적으므로 속도는 비교적 느려도 된다.
중기 스케줄러 (medium term scheduler)	• 주기억장치에서 CPU를 점유하기 위해 서로 경쟁하는 프로세스의 수를 줄여서 다중 프로그래밍의 정도를 완화시킨다. (swapping 기법의 일부) • 시분할 시스템의 운영체제에서 주로 사용되는 것이다.

(1) 선입선출(FCFS;First Come First Served) 스케줄링 - 비선점

- FCFS(first come first served)는 준비큐에 도착한 프로세스 순으로 CPU를 할당한다.
- FCFS는 큐를 이용하여 쉽게 구현할 수 있다. - FIFO(First in First out)
- FCFS는 비선점 방식으로 **대화식 시스템에 부적합**하다.
- FCFS는 일정시간 간격으로 CPU를 사용하는 **시분할 시스템에 부적합**하다.
- FCFS는 **중요한 작업이 중요하지 않은 긴 작업 때문에 기다려야 하는 문제점**이 있다.

[예] 준비 큐에 p1, p2, p3 순으로 시간 0에 도착한 프로세스 집합

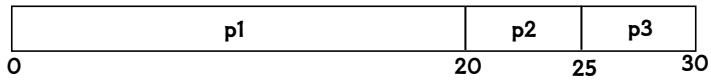
(시간 단위 : 밀리초)

프로세스	Burst 시간	대기시간	반환시간
p1	20	0 - 0 = 0	20
p2	5	20 - 0 = 20	25
p3	5	25 - 0 = 25	30

// Burst 시간
- Burst 시간은 프로세스가 작업을 완료하는데 필요한 시간이다.
- Burst는 자료를 전송한 단위로 취급되는 연속 신호의 개념이다.

- **대기시간 = 다른 프로세스가 작업한 시간 - 도착시간**
- **반환시간 = Burst 시간 + 대기시간**

// 이를 간트 차트(gantt chart)로 그리면 다음과 같다.



- 평균대기시간 = $\frac{0+20+25}{3} = 15(\text{밀리초})$
- 평균반환시간 = $\frac{20+25+30}{3} = 25(\text{밀리초})$

[결론] 선입선출 스케줄링은 비선점 기법이므로 긴 작업시간을 가지는 프로세스가 먼저 도착하게 되면 평균대기시간과 평균반환시간이 길어지게 되고 CPU 이용률은 저하된다.

(2) 최단작업우선(SJF; Shortest Job First) 스케줄링 - 비선점

- SJF는 작업시간이 가장 짧은 프로세스에게 CPU를 우선 할당한다.
- SJF는 **평균대기시간을 최소**로 만드는 것을 목표로 두고 있는 알고리즘이다.
- 두 프로세스가 같은 작업시간을 가지면, 순서 결정을 위해 선입선출 알고리즘을 적용한다.
- 작업시간이 긴 프로세스는 **기아상태**가 발생할 수 있다.
- SJF 알고리즘은 선점 또는 비선점 방식으로 적용될 수 있다.
- 선점 방식으로 적용되는 SJF 알고리즘을 특별히 SRTF 스케줄링이라 한다.

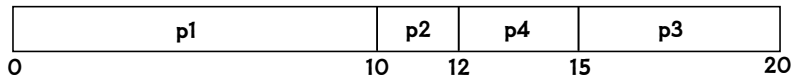
[예] 준비 큐에 다음처럼 프로세스가 도착한 경우

(시간 단위 : 밀리초)

프로세스	도착시간	Burst 시간	대기시간	반환시간
p1	0	10	0 - 0 = 0	10 + 0 = 10
p2	1	2	10 - 1 = 9	2 + 9 = 11
p3	2	5	15 - 2 = 13	5 + 13 = 18
p4	3	3	12 - 3 = 9	3 + 9 = 12

- **대기시간 = 다른 프로세스가 작업한 시간 - 도착시간**
- **반환시간 = Burst 시간 + 대기시간**

// 이를 간트 차트(gantt chart)로 그리면 다음과 같다.



- 평균대기시간 = $\frac{0+9+13+9}{4} = 7.75(\text{밀리초})$
- 평균반환시간 = $\frac{10+11+18+12}{4} = 12.75(\text{밀리초})$

◈ **SJF 스케줄링 분석**

- 모든 프로세스가 동시에 도착하면, SJF 스케줄링은 비선점 기법으로 처리하여도 프로세스 집합에 대해 최소의 평균대기시간을 가지게 된다.
- 각 프로세스 도착시간이 서로 다르면, 비선점 SJF는 최소의 평균대기시간을 가질 수 없다.
- 비선점 SJF 스케줄링은 긴 작업이 먼저 도착할수록 평균대기시간은 길어진다.
- SJF 스케줄링은 모든 프로세스의 작업시간을 파악하는데 어려움이 있다.
- 해서, SJF 스케줄링은 단기 스케줄링에서는 구현하기 어렵고, 주로 장기 스케줄링에 사용된다.

(3) 최소잔여시간우선(SRT; Shortest Remaining Time-first) 스케줄링 - 선점

- SRT는 비선점형 SJF 알고리즘에 **선점 기법을 추가한 것**이다.
- SRT는 새로 도착한 프로세스가 실행중인 프로세스의 남아 있는 작업시간보다 더 짧은 작업시간을 가지면 CPU를 **선점**하도록 한다.

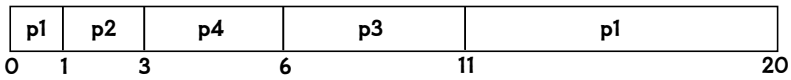
[예] 준비 큐에 다음처럼 프로세스가 도착한 경우

(시간 단위 : 밀리초)

프로세스	도착시간	Burst 시간	대기시간	반환시간
p1	0	10	10 - 0 = 10	10 + 10 = 20
p2	1	2	1 - 1 = 0	2 + 0 = 2
p3	2	5	6 - 2 = 4	5 + 4 = 9
p4	3	3	3 - 3 = 0	3 + 0 = 3

- **대기시간 = 다른 프로세스가 작업한 시간 - 도착시간**
- **반환시간 = Burst 시간 + 대기시간**

// 이를 간트 차트(gantt chart)로 그리면 다음과 같다.



- 평균대기시간 = $\frac{10+0+4+0}{4} = 3.5$(밀리초)
- 평균반환시간 = $\frac{20+2+9+3}{4} = 8.5$(밀리초)

결론	SRT 스케줄링은 선점 기법이므로 긴 작업시간을 가지는 프로세스가 준비큐에 먼저 도착하여도 평균대기시간과 평균반환시간은 최소값을 가지게 된다. CPU를 강제로 빼앗을 수 있기 때문이다.

[Tip] SRT 스케줄링

SRT 스케줄링을 '선점형 SJF 스케줄링'이라고도 한다.

(4) 우선순위(priority) 스케줄링 - 선점 또는 비선점

- 각 프로세스에 중요도에 따라 우선순위를 부여한다.
- 우선순위는 중요도에 따라 내부적, 외부적으로 정의될 수 있다.
- 우선순위가 가장 높은 프로세스에게 CPU를 할당한다.
- 우선순위가 낮은 프로세스는 **기아상태**에 빠질 수 있다.

〈노화(aging) 기법〉

우선순위가 낮은 프로세스들이 무한히 봉쇄되는 것을 방지하기 위해 우선순위를 점진적으로 높이는 방안 → 일정시간이 지나면 우선순위를 1씩 증가시킨다.

(5) 기한부(deadline) 스케줄링 - 비선점

- 작업이 주어진 시간(deadline) 안에 완료되도록 하는 기법이다.
- 사용자는 deadline 정보를 제공해야 하고,
- 시스템은 deadline을 지키기 위한 계획을 세워야 한다.

(6) HRN(Highest Response ratio Next) 스케줄링 - 비선점

- HRN은 SJF 기법의 단점인 긴 작업과 짧은 작업의 불평등을 보완한 기법이다.
- HRN은 우선순위를 다음 공식으로 결정한다.

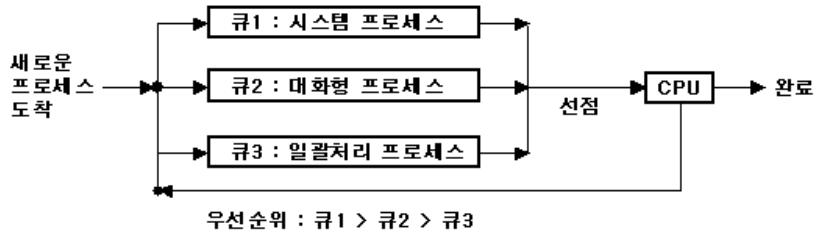
- **HRN 우선순위** = $\frac{\text{대기시간} + \text{작업시간(서비스시간)}}{\text{작업시간(서비스시간)}}$

  ㄴ, 긴 작업도 오래 대기하면 우선순위가 높아진다.

(7) RR(Round Robin) 스케줄링 - 선점

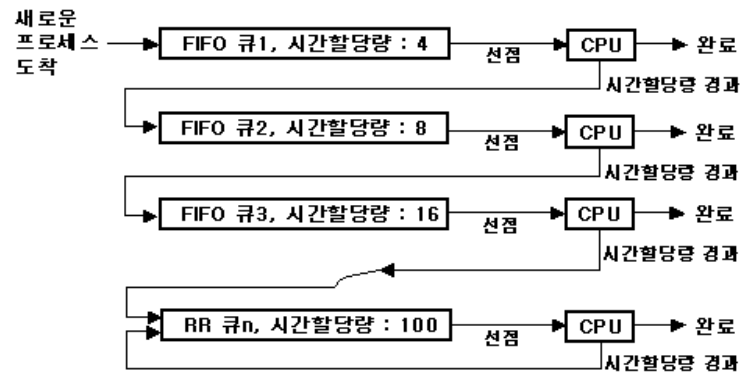
- RR은 **시분할 시스템**을 구현하기 위한 알고리즘이다.
- RR은 **시간할당량**(time quantum)이라는 고유한 CPU 할당시간을 미리 정의한다.
- RR은 어떤 프로세스에게도 시간할당량 이상은 CPU를 할당하지 않는다.
- RR은 준비큐를 **원형큐로 설계**한다.
- RR은 프로세스들 사이에 우선순위를 두지 않는다.**(첫 번째 프로세스부터 차례로 CPU를 할당)**
- RR은 할당된 시간 내에 작업을 끝내지 못한 프로세스는 **준비큐의 맨 뒤에 배치**한다.
- RR에서 **시간할당량이 매우 크면**, RR은 FCFS와 같고
- RR에서 **시간할당량이 매우 적으면**, n개의 프로세스는 각각 CPU를 $\frac{1}{n}$만큼씩 사용하게 된다.
- 즉, 각 프로세스는 자신의 고유한 처리기를 가지고 있는 것처럼 보인다.**(CPU 공유)**

(8) 다단계 큐(multilevel queue) 스케줄링 - 선점



- 다단계 큐는 **프로세스 특징**별로 여러 그룹으로 나누고, 여러 준비큐를 이용하는 기법이다.
- 다단계 큐에서 각 큐는 **자신 고유의 스케줄링** 알고리즘을 가진다.
- 다단계 큐는 다중처리에서 사용되는 Foreground와 Background 개념을 사용한다.
- 포그라운드 큐는 백그라운드 큐보다 높은 우선순위를 가진다.
- 포그라운드 큐는 RR, 백그라운드 큐는 FCFS으로 구현될 수 있다.
- 다단계 큐는 큐1과 큐2가 비어 있을 경우만 큐3에 있는 프로세스가 CPU를 할당받는다.

(9) 다단계 피드백 큐(multilevel feedback queue) 스케줄링 - 선점



- **다단계 피드백 큐** 스케줄링에서는 프로세스가 **큐 사이를 이동**할 수 있다.
  → **다단계 큐**에서는 프로세스가 **하나의 큐에 영구적으로 할당**된다.(융통성 부족)
- 다단계 피드백 큐에서 **최하위 큐는 가장 긴 시간할당량**을 가진다.
- 상위 큐는 하위 큐보다 짧은 시간할당량을 가진다.(상위 큐는 하위 큐를 선점)
- 다단계 피드백 큐는 **입출력 위주의 프로세스에게 우선권**을 부여한다.**(시스템 성능 향상)**

↓ 시스템 성능이 향상되는 이유

- 입출력 프로세스가 짧은 시간동안 CPU를 이용하여 연산한 후, 입출력이 발생되면
- 입출력 프로세스는 CPU를 다른 프로세스에게 양보한다.(다른 프로세스에게 CPU 할당)
- 즉, 입출력과 CPU 작업을 병행 진행할 수 있다.

▣ 탐구 – 시스템 성능 향상

// 프로세스 종류

입출력 위주 프로세스	• 입출력장치를 많이 이용하는 프로세스 • 입출력 위주의 프로세스는 일반적으로 **대화형 프로세스**이다.
CPU 위주 프로세스	• CPU를 많이 이용하는 프로세스 • **연산 위주의 프로세스**라고도 한다.
상호 작용 프로세스	• 사용자와의 인터페이스가 많은 프로세스
일괄 작업 프로세스	• 사용자와의 인터페이스를 필요로 하지 않은 프로세스
실시간 프로세스	• 시간 지연 없이 즉시 수행되는 프로세스
협력 프로세스	• 실행 중에 서로 영향을 주고받는 프로세스 • 코드나 데이터 등 자원을 공유하게 된다.

◉ 시스템 성능 향상을 위한 선점 스케줄링의 고려 사항

① **CPU 위주**의 프로세스보다 **입출력 위주의 프로세스**에 **높은 우선순위를 부여**한다.
- 입출력 위주의 프로세스를 먼저 스케줄링하면
- 입출력 위주의 프로세스는 CPU를 조금만 사용하고 입출력을 위해 CPU를 양보한다.
- 결과적으로, CPU와 입출력장치를 동시 사용하므로 자원 활용도가 증가하게 된다.
- **입출력 위주의 프로세스**는 일반적으로 **대화형 프로세스**이다.
- 대화형 프로세스는 빠른 응답을 원하는 경우가 많으므로 높은 우선순위를 부여한다.
- 다단계 피드백 큐 스케줄링에서 입출력 위주의 프로세스에게 우선권을 부여하는 이유

② **실시간 프로세스**는 높은 우선순위를 부여한다.
- 실시간 프로세스의 실행이 늦으면 재앙적 상황이 발생할 수 있으므로

③ 타임 슬라이스는 프로세스들의 CPU 평균반환시간보다 조금 크게 책정한다.
- 불필요한 문맥교환에 따른 오버헤드를 줄이기 위해

기출문제 분석

1. 다음 〈보기〉 중 프로세스 스케줄링을 선점 스케줄링과 비선점 스케줄링으로 구분한 것으로 옳은 것은? [2019년 국회 9급]

------<보기>----------

ㄱ. RR(Round Robin)　　ㄴ. SJF(Shortest Job First)

ㄷ. SRT(Shortest Remaining Time)　　ㄹ. HRN(Highest Response ratio Next)

ㅁ. MFQ(Multilevel Feedback Queue)　　ㅂ. MLQ(MultiLevel Queue)

	선점 스케줄링	비선점 스케줄링
①	ㅁ, ㅂ	ㄱ, ㄴ, ㄷ, ㄹ
②	ㄱ, ㄴ, ㄷ	ㄹ, ㅁ, ㅂ
③	ㄱ, ㅁ, ㅂ	ㄴ, ㄷ, ㄹ
④	ㄱ, ㄴ, ㄷ, ㄹ	ㅁ, ㅂ
⑤	ㄱ, ㄷ, ㅁ, ㅂ	ㄴ, ㄹ

♤ **선점 스케줄링 / 비선점 스케줄링**

- 선점 스케줄링 : RR, SRT, MFQ, MLQ
- 비선점 스케줄링 : SJF(Shortest Job First), HRN(Highest Response ratio Next)

정답 : ⑤

2. 하나의 프로세스가 CPU를 할당받은 후에는, 스스로 CPU를 반납할 때까지 다른 프로세스가 CPU를 차지할 수 없는 스케줄링 기법에 해당하는 것만을 모두 고르면? [2019년 지방 9급]

ㄱ. FCFS(first come first served)

ㄴ. RR(round robin)

ㄷ. SRT(shortest remaining time)

---

① ㄱ　　② ㄱ, ㄷ　　③ ㄴ, ㄷ　　④ ㄱ, ㄴ, ㄷ

♤ **스케줄링 기법 - 선점/비선점**

ㄱ. FCFS(first come first served) → 비선점

ㄴ. RR(round robin) → 선점

ㄷ. SRT(shortest remaining time) → 선점

정답 : ①

3. 운영체제의 스케줄링 기법에 대한 설명으로 옳지 않은 것은? [2017년 법무 9급]

① FCFS(First-Come-First-Served) 스케줄링은 비선점(nonpreemptive) 방식으로 실행 중인 프로세스가 종료하면 준비 큐에서 가장 오래 대기한 프로세스를 다음 실행 프로세스로 선정한다.

② RR(Round-Robin) 스케줄링은 선점(preemptive) 방식으로 프로세스를 정해진 시간 할당량만큼 실행 후 종료하지 못하면 준비 큐로 이동시킨다.

③ 비선점 SJF(Shortest-Job-First) 스케줄링은 준비 큐에서 예상 전체 실행시간이 가장 짧은 프로세스를 다음 실행 프로세스로 선정한다.

④ 선점 SJF 스케줄링은 SRTF(Shortest-Remaining-Time-First) 스케줄링이라고 불리며 비선점 SJF 스케줄링에서 발생할 수 있는 기아상태(starvation) 문제를 해결한다.

♤ **운영체제의 스케줄링**

- 선점 SJF 스케줄링은 SRTF(Shortest-Remaining-Time-First) 스케줄링이라고 불리며 비선점 SJF 스케줄링에서 발생할 수 있는 기아상태(starvation) 문제를 해결한다.**(×)**
  → SRTF는 **기아상태**가 발생할 수 있다.(짧은 작업이 긴 작업을 선점하므로)

정답 : ④

4. 어떤 프로세스가 일정 크기의 CPU 시간 할당량(time quantum)을 한 번 받은 후에는 강제로 대기 큐의 다른 프로세스에게 CPU를 넘겨주는 방식의 스케줄링 기법은? [2016년 지방 9급]

① FCFS(First-Come-First-Served)

② RR(Round-Robin)

③ SPN(Shortest Process Next)

④ HRRN(Highest Response Ratio Next)

♤ **RR(Round-Robin) – 선점**

- RR은 **시분할 시스템**을 구현하기 위한 알고리즘이다.
- RR은 **시간할당량**(time quantum)이라는 고유한 CPU 할당시간을 미리 정의한다.
- RR은 어떤 프로세스에게도 시간할당량 이상은 CPU를 할당하지 않는다.
- RR은 준비큐를 **원형큐로 설계**한다.
- RR은 프로세스들 사이에 우선순위를 두지 않는다.**(첫 번째 프로세스부터 차례로 CPU를 할당)**
- RR은 할당된 시간 내에 작업을 끝내지 못한 프로세스는 **준비큐의 맨 뒤에 배치**한다.

정답 : ②

5. 프로세스 P가 수행 준비는 되어 있으나 다른 프로세스들이 더 우선적으로 수행되어, 프로세스 P가 계속적으로 CPU 할당을 기다리면서 수행되지 못하는 상태는? [2017년 경기 추가 9급]

① 교착상태(deadlock) ② 기아상태(starvation)
③ 경쟁상태(race condition) ④ 상호배제(mutual exclusion)

♤ **교착상태와 기아상태 비교**

구분	교착상태(deadlock)	기아상태(starvation)
정의	여러 프로세스가 아무 일도 하지 못하고 특정 사건이 발생되기를 무한정 대기	특정 프로세스가 원하는 자원(CPU, 디스크 등)을 할당 받기 위해서 무한정 대기
발생 원인	상호배제, 비선점, 점유와 대기, 순환대기	자원의 **편중**된 분배 정책
해결 방안	예방, 회피(은행가 알고리즘), 발견, 회복	노화(aging) 기법, 자원의 공평한 분배

• 교착상태는 하나 이상의 작업에 영향을 끼치므로 기아상태보다 문제가 더 심각하다.
• 교착상태가 프로세스에게 한정된 자원을 자유롭게 할당한 결과라면
• 기아상태는 프로세스에게 한정된 자원을 편중되게 할당한 결과이다.

정답 : ②

6. 다음 〈보기〉의 CPU 스케줄링 기법 중 기아(starvation) 현상이 발생할 수 있는 스케줄링 기법으로 옳은 것을 모두 고르면? [2017년 국회 9급]

ㄱ. FIFO(First In First Out) 스케줄링
ㄴ. RR(Round Robin) 스케줄링
ㄷ. Priority 스케줄링
ㄹ. HRN(Highest Response ratio Next) 스케줄링
ㅁ. SJF(Shortest Job First) 스케줄링
---

① ㅁ ② ㄱ, ㄴ ③ ㄷ, ㅁ
④ ㄴ, ㄷ, ㄹ ⑤ ㄷ, ㄹ, ㅁ

♤ **CPU 스케줄링 기법 중 기아 현상**

ㄷ. Priority 스케줄링 : 우선순위가 낮은 프로세스는 기아상태'가 될 수 있다.
→ 이를 해결하기 위한 것이 노화(Aging) 기법이다.
ㅁ. SJF(Shortest Job First) 스케줄링 : 실행시간이 긴 작업은 기아상태에 빠질 가능성이 있다.

정답 : ③

7. CPU 스케줄링 기법 중에서 기아상태(starvation)가 발생할 가능성이 없는 것만을 모두 고르면? [2014년 지방 9급]

> ㄱ. FCFS(First-Come First-Served)
> ㄴ. 라운드 로빈(RR : Round Robin)
> ㄷ. SJF(Shortest Job First)
> ㄹ. HRRN(Highest Response Ratio Next)

① ㄱ, ㄴ ② ㄷ, ㄹ ③ ㄱ, ㄴ, ㄷ ④ ㄱ, ㄴ, ㄹ

♤ **CPU 스케줄링 – 기아상태 발생**

ㄱ. FCFS(First-Come First-Served) → 도착한 프로세스 순으로 CPU 할당
ㄴ. 라운드 로빈(RR : Round Robin) → 프로세스에게 시간 할당량만큼만 CPU 할당
ㄷ. SJF(Shortest Job First) → 긴 작업의 프로세스는 '**기아상태**'에 빠질 수 있다.
ㄹ. HRRN(Highest Response Ratio Next) → 긴 작업과 짧은 작업의 불평등 보완

정답 : ④

8. 다음 중 입출력 위주의 프로세스와 연산 위주의 프로세스의 특성에 따라 CPU 사용시간(할당량)을 다르게 부여하는 선점 방식의 CPU 스케줄링 기법으로 옳은 것은? [2016년 국회 9급]

① Round-Robin ② Multi-level Feedback Queue
③ Shortest Job First ④ Highest Response ratio Next
⑤ Deadline

♤ **다단계 피드백 큐(Multilevel Feedback Queue) 스케줄링 – 선점**

- **다단계 피드백 큐** 스케줄링에서는 프로세스가 **큐 사이를 이동**할 수 있다.
  → **다단계 큐**에서는 프로세스가 **하나의 큐에 영구적으로 할당**된다.(융통성 부족)
- 다단계 피드백 큐에서 **최하위 큐는 가장 긴 시간할당량**을 가진다.
- 상위 큐는 하위 큐보다 짧은 시간할당량을 가진다.(상위 큐는 하위 큐를 선점)
- 다단계 피드백 큐는 **입출력 위주의 프로세스에게 우선권**을 부여한다.**(시스템 성능 향상)**

↓시스템 성능이 향상되는 이유

- 입출력 프로세스가 짧은 시간동안 CPU를 이용하여 연산한 후, 입출력이 발생되면
- 입출력 프로세스는 CPU를 다른 프로세스에게 양보한다.(다른 프로세스에게 CPU 할당)
- 즉, 입출력과 CPU 작업을 병행 진행할 수 있다.

정답 : ②

9. 프로세스 P1, P2, P3, P4를 선입선출(First In First Out) 방식으로 스케줄링을 수행할 경우 평균응답시간으로 옳은 것은? (단, 응답시간은 프로세스 도착시간부터 처리가 종료될 때까지의 시간을 말한다) [2018년 계리직]

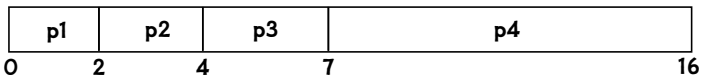
프로세스	도착시간	처리시간
P1	0	2
P2	2	2
P3	3	3
P4	4	9

① 3 ② 4 ③ 5 ④ 6

♤ 선입선출(first in first out, first come first served) 방식 – 비선점

- FCFS(first come first served)는 준비큐에 도착한 프로세스 순으로 CPU를 할당한다.
- FCFS는 비선점 방식으로 **대화식 시스템에 부적합**하다.
- FCFS는 일정시간 간격으로 CPU를 사용하는 **시분할 시스템에 부적합**하다.
- FCFS는 **중요한 작업이 중요하지 않은 긴 작업 때문에 기다려야 하는 문제점**이 있다.

// 간트 차트(gantt chart)로 그리면 다음과 같다.



↓ 표로 정리하면

프로세스	도착시간	처리시간	대기시간	응답시간
P1	0	2	0 – 0 = 0	2 + 0 = 2
P2	2	2	2 – 2 = 0	2 + 0 = 2
P3	3	3	4 – 3 = 1	3 + 1 = 4
P4	4	9	7 – 4 = 3	9 + 3 = 12

- **대기시간 = 다른 프로세스가 작업한 시간 – 도착시간**
- **응답시간 = 처리시간 + 대기시간**

- 평균대기시간 = $\frac{0+0+1+3}{4}$ = 1
- 평균응답시간 = $\frac{2+2+4+12}{4}$ = **5**

정답 : ③

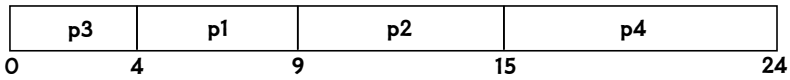
10. 다음과 같이 P1, P2, P3, P4 프로세스가 동시에 준비 상태 큐에 도착했을 때 SJF(Shortest Job First) 스케줄링 알고리즘에서 평균반환시간과 평균대기시간을 바르게 연결한 것은? (단, 프로세스 간 문맥교환에 따른 오버헤드는 무시하며, 주어진 4개의 프로세스 외에 처리할 다른 프로세스는 없다고 가정한다) [2022년 지방 9급]

프로세스	실행시간
P1	5
P2	6
P3	4
P4	9

	평균반환시간	평균대기시간
①	6	6
②	6	7
③	13	6
④	13	7

♤ SJF(Shortest Job First) 스케줄링

// 간트 차트를 그리면 다음과 같다.



프로세스	실행시간	대기시간	반환시간
P1	5	4 - 0 = 4	5 + 4 = 9
P2	6	9 - 0 = 9	6 + 9 = 15
P3	4	0 - 0 = 0	4 + 0 = 4
P4	9	15 - 0 = 15	9 + 15 = 24

- **대기시간 = 다른 프로세스가 작업한 시간 − 도착시간**
- **반환시간 = 실행시간 + 대기시간**

- 평균대기시간 $= \frac{4+9+0+15}{4} = 7$
- 평균반환시간 $= \frac{9+15+4+24}{4} = 13$

정답 : ④

11. 다음 표는 단일 중앙처리장치에 진입한 프로세스의 도착시간과 그 프로세스를 처리하는 데 필요한 실행시간을 나타낸 것이다. 비선점 SJF(Shortest Job First) 스케줄링 알고리즘을 사용한 경우, P1, P2, P3, P4 프로세스 4개의 평균대기시간은? (단, 프로세스간 문맥교환에 따른 오버헤드는 무시하며, 주어진 4개의 프로세스 외에 처리할 다른 프로세스는 없다고 가정한다) [2018년 지방 9급]

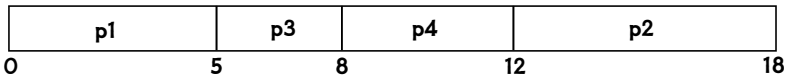
프로세스	도착시간(ms)	실행시간(ms)
P1	0	5
P2	3	6
P3	4	3
P4	6	4

① 3 ms ② 3.5 ms
③ 4 ms ④ 4.5 ms

♤ 비선점 SJF(Shortest Job First) 스케줄링

- SJF는 작업시간이 가장 짧은 프로세스에게 CPU를 할당한다.
- SJF는 평균대기시간을 최소로 만드는 것을 목표로 두고 있는 알고리즘이다.
- 두 프로세스가 같은 작업시간을 가지면, 순서 결정을 위해 선입선출 알고리즘을 적용한다.
- 작업시간이 긴 프로세스는 **기아상태**가 발생할 수 있다.

// 간트차트로 그리면 다음과 같다.



↓ 표로 정리하면

프로세스	도착시간(ms)	실행시간(ms)	대기시간	실행순서
P1	0	5	0 - 0 = 0	①
P2	3	6	12 - 3 = 9	④
P3	4	3	5 - 4 = 1	②
P4	6	4	8 - 6 = 2	③

- **대기시간 = 다른 프로세스가 작업한 시간 − 도착시간**
- **응답시간 = 처리시간 + 대기시간**

- 평균대기시간 = $\frac{0+9+1+2}{4}$ = 3 (ms)

정답 : ①

12. 다음 표는 프로세스들의 대기시간과 예상되는 서비스 시간을 나타낸 것이다. HRRN(Highest Response Ratio Next) 스케줄링 알고리즘을 사용할 때, 우선순위가 가장 높은 프로세스는? [2017년 경기 추가 9급]

프로세스	대기시간	서비스 시간
P1	10	5
P2	12	4
P3	8	12
P4	15	3

(단위:밀리초)

① P1 ② P2 ③ P3 ④ P4

♤ HRRN에서 우선순위 = (대기시간 + 서비스 시간) / 서비스 시간

- P1 우선순위 = (10 + 5) / 5 = 3
- P2 우선순위 = (12 + 4) / 4 = 4
- P3 우선순위 = (8 + 12) / 12 = 1.666667
- P4 우선순위 = (15 + 3) / 3 = 6 → 이 값이 클수록 먼저 처리된다.(높은 우선순위)

정답 : ④

13. 다음 프로세스 집합에 대하여 라운드 로빈 CPU 스케줄링 알고리즘을 사용할 때, 프로세스들의 총 대기시간은? (단, 시간 0에 P1, P2, P3 순서대로 도착한 것으로 하고, 시간 할당량은 4밀리초로 하며, 프로세스 간 문맥교환에 따른 오버헤드는 무시한다) [2017년 국가 9급]

프로세스	버스트 시간(밀리초)
P1	30
P2	3
P3	4

① 16 ② 18 ③ 20 ④ 24

♤ 라운드 로빈 CPU 스케줄링 : 시간 할당량은 4밀리초

프로세스	버스트 시간(밀리초)	대기시간
P1	30	7 - 0 = 7
P2	3	4 - 0 = 4
P3	4	7 - 0 = 7

- 대기시간 = 다른 프로세스가 작업한 시간 - 도착시간
- 프로세스들의 총 대기시간 = 7 + 4 + 7 = 18

정답 : ②

14. 〈보기〉의 프로세스 P1, P2, P3을 시간 할당량(time quantum)이 2인 RR(Round-Robin) 알고리즘으로 스케줄링할 때, 평균응답시간으로 옳은 것은? (단, 응답시간이란 프로세스의 도착시간부터 처리가 종료될 때까지의 시간을 말한다. 계산 결과값을 소수점 둘째자리에서 반올림한다) [2016년 계리직]

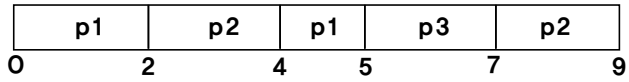
프로세스	도착시간	실행시간
P1	0	3
P2	1	4
P3	3	2

① 5.7 ② 6.0 ③ 7.0 ④ 7.3

♤ RR 알고리즘(선점) – 어려운 문제(?)

• RR은 프로세스들 사이에 우선순위를 두지 않고, 순서대로 CPU를 할당하는 방식이다.

• 간트 차트(gantt chart)로 그리면 다음과 같다.**(시간할당량 2)**



↓큐 상태 변화

p1	p1 큐에 도착, p1 실행
p1, p2	p2 큐에 도착. p1 계속 실행 중
p2, p1	p1은 시간할당량 2만큼 사용하고 큐 뒤로 감**(p1의 남은 시간은 1)**. p2 실행
p2, p1, p3	p3 큐에 도착. p2는 계속 실행 중
p1, p3, p2	p2가 시간할당량 2만큼 사용하고 큐 뒤로 감**(p2의 남은 시간은 2)**. p1 실행
p3, p2	p1은 시간 1만큼 사용하고 작업 종료**(큐에 있을 필요가 없음)**, p3 실행
p2	p3은 시간할당량 2만큼 사용하고 작업 종료**(큐에 있을 필요가 없음)**, p2 실행

↓표로 정리하면 다음과 같다.

프로세스	도착시간	실행시간	대기시간	응답시간
P1	0	3	2 - 0 = 2	2 + 3 = 5
P2	1	4	5 - 1 = 4	4 + 4 = 8
P3	3	2	5 - 3 = 2	2 + 2 = 4

• 대기시간 = 다른 프로세스의 총 작업시간 – 늦게 도착한 시간

• 응답시간 = 대기시간 + 실행시간

• 평균응답시간 = $\frac{5+8+4}{3}$ = 5.6666... = **5.7**

정답 : ①

15. 다음과 같이 3개의 프로세스가 있다고 가정한다. 각 프로세스의 도착시간과 프로세스의 실행에 필요한 시간은 아래 표와 같다. CPU 스케줄링 알고리즘으로 RR(Round Robin)을 사용한다고 가정한다. 3개의 프로세스가 CPU에서 작업을 하고 마치는 순서는? (단, CPU를 사용하는 타임 슬라이스(time slice)는 2이다) [2017년 서울 9급]

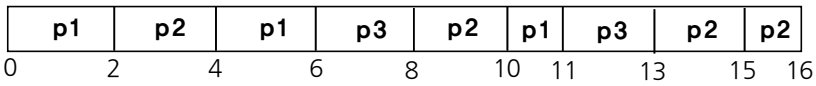
프로세스	도착시간	프로세스의 실행에 필요한 시간
P1	0	5
P2	1	7
P3	3	4

① P2, P1, P3　② P2, P3, P1
③ P1, P2, P3　④ P1, P3, P2

♤ CPU 스케줄링 – RR(Round Robin)

- RR은 프로세스들 사이에 우선순위를 두지 않고, 순서대로 CPU를 할당하는 방식이다.
- RR은 **시분할 시스템**을 구현하기 위한 알고리즘이다.
- RR은 **시간할당량**(time quantum)이라는 고유한 CPU 할당시간을 미리 정의한다.
- RR은 준비큐를 **원형큐로 설계**한다. RR은 첫 번째 프로세스부터 차례로 CPU를 할당한다.

// 간트 차트(gantt chart)로 그리면 다음과 같다.(시간할당량 2)



↓큐 상태 변화

p1	p1 큐에 도착, p1 실행
p1, p2	p2 큐에 도착. p1 계속 실행 중
p2, p1	p1은 시간할당량 2만큼 사용하고 큐 뒤로 감. p2 실행 (p1은 3시간 남음)
p2, p1, p3	p3 큐에 도착. p2 계속 실행 중
p1, p3, p2	p2가 시간할당량 2만큼 사용하고 큐 뒤로 감. p1 실행 (p2는 5시간 남음)
p3, p2, p1	p1은 시간할당량 2만큼 사용하고 큐 뒤로 감. p3 실행 (p1은 1시간 남음)
p2, p1, p3	p3가 시간할당량 2만큼 사용하고 큐 뒤로 감. p2 실행 (p3은 2시간 남음)
p1, p3, p2	p2가 시간할당량 2만큼 사용하고 큐 뒤로 감 (p2는 3시간 남음)
p1, p3, p2	p1 실행. p1은 남은 작업 1 시간을 실행하고 종료 **(p1 작업 종료)**
p3, p2	p3 실행, p3은 남은 작업 2 시간을 실행하고 종료 **(p3 작업 종료)**
p2	p2 실행, p2가 시간할당량 2만큼 사용 (p2는 1시간 남음)
p2	p2 실행, p2가 남은 작업 1 시간을 실행하고 종료 **(p2 작업 종료)**

정답 : ④