운 영 체 제

2장 프로세스 관리(2)

학습 내용

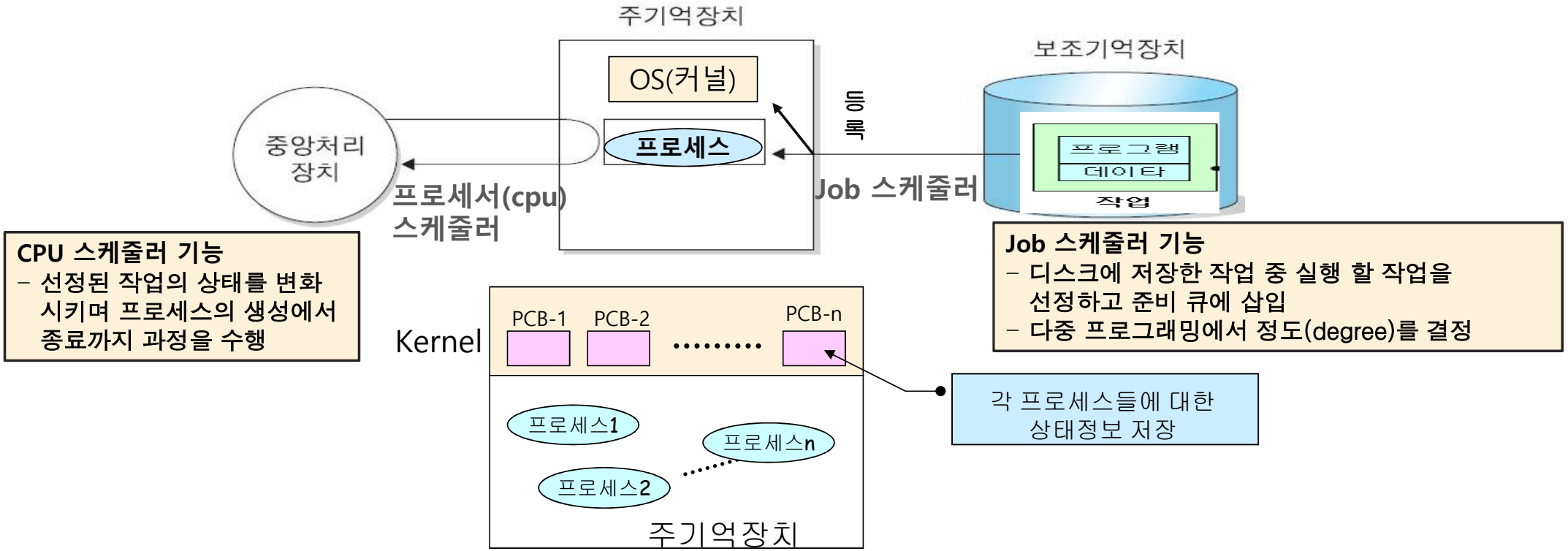
* 2장 프로세스 관리
  * 시스템 소프트웨어의 종류와 기능 <= 2장 프로세스 관리와 5장 메모리 관리 관련
    * 컴파일러와 인터프리터
    * 어셈블리어와 어셈블러
    * 링커와 로더
  * 프로세스 개요
    * 명령어 실행 과정
    * 프로세스 개념
    * 프로세스 정의
    * 프로세스 제어 블록(PCB)
    * 프로세스의 상태
    * 프로세스 상태의 전이
    * 프로세스 상태의 전이 정리
  * 프로세스 스케줄링 개요
    * 프로세스 스케줄링 개요
    * 비선점 스케줄링
    * 선점 스케줄링

2장 프로세스 관리(2)

시스템 소프트웨어의 종류와 기능
프로세스 개요
프로세스 스케줄링 개요

스케줄링의 개요

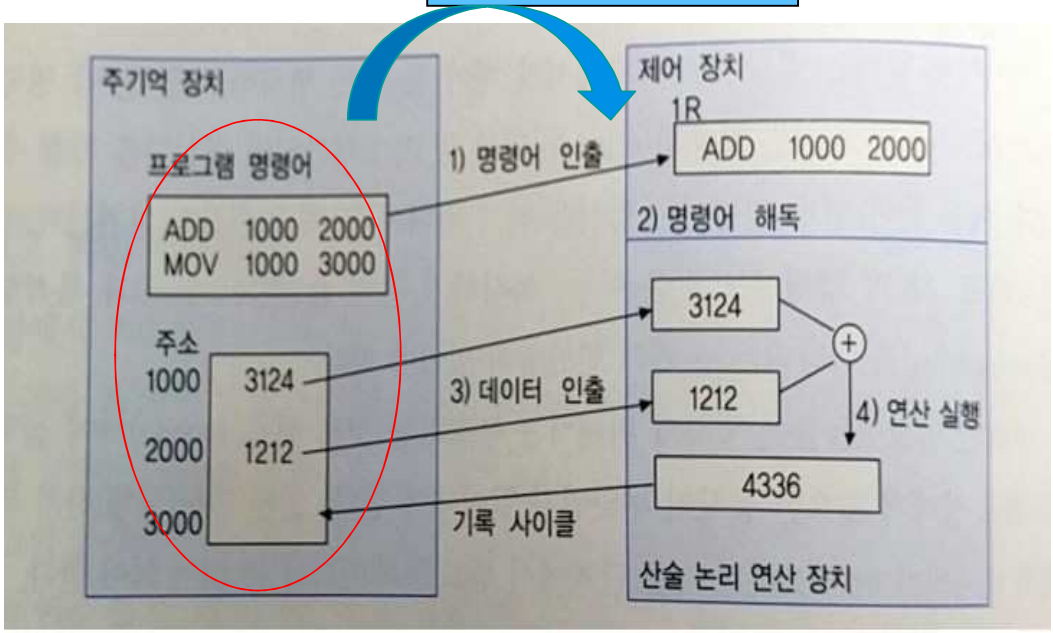
- **스케줄링 개념**
  - **스케줄링은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미한다.**
  - **프로세스가 생성되어 완료될 때까지 프로세스는 여러 종류의 스케줄링 과정을 거치게 된다.**

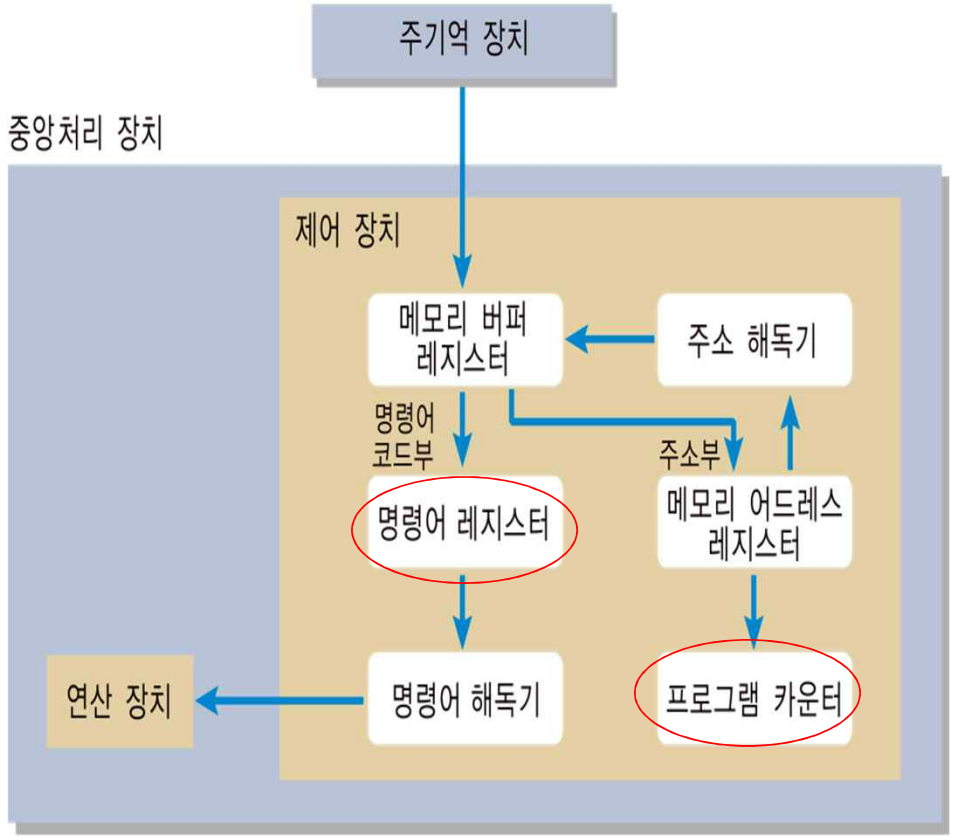


스케줄링의 개요

- 컴퓨터 동작 (예: 명령어 실행 과정)
  1. 명령어 인출(메모리로부터 명령어 가져오기)
  2. 명령어 해독
  3. 데이터 인출
  4. 실행

OS
-프로세서 스케줄러
-디스패처





IR(명령어 레지스터), PC(프로그램 카운터), MBR(메모리 버퍼 레지스터), MAR(메모리 어드레스 레지스터)

스케줄링의 목적

- 스케줄링의 목적(CPU나 자원을 효율적으로 사용하기 위한 정책)
  - 공정성 : 모든 프로세스에 공정하게 할당한다.
  - 처리율증가 : 단위 시간당 프로세스를 처리하는 비율을 증가시킨다.
  - CPU이용률 증가 : 프로세스 실행과정에서 주기억장치를 액세스 한다든지 입/출력 실행등의 원인에 의해 발생할 수 있는 CPU낭비시간을 줄이고, CPU가 순수하게 프로세스를 실행하는데 사용되는 시간 비율을 증가시킨다.
  - 우선순위 제도 : 우선순위가 높은 프로세스를 먼저 실행한다.
  - 오버헤드 최소화 : 오버헤드를 최소화한다.
  - 응답시간 최소화 : 작업을 지시하고, 반응하기 시작하는 시간을 최소화한다.
  - 반환시간 최소화 : 프로세스를 제출한 시간부터 실행이 완료될때까지 걸리는 시간을 최소화한다.
  - 대기시간 최소화 : 프로세스가 준비상태 큐에서 대기하는 시간을 최소화한다.
  - 균형있는 자원의 사용 : 메모리, 입/출력장치 등의 자원을 균형있게 사용한다.
  - 무한 연기 회피 : 자원을 사용하기 위해 무한정 연기되는 상태를 회피한다.

스케줄링의 기준

- 스케줄링의 기준
  - 입출력 위주의 프로세스인가?
  - 연산 위주의 프로세스인가?
  - 프로세스가 일괄 처리 형인가 대화형인가?
  - 긴급한 응답이 요구되는가?
  - 프로세스의 우선순위
  - 프로세스가 페이지 부재를 얼마나 자주 발생시키는가?
  - 높은 우선순위를 지니는 프로세스에 의해서 얼마나 자주 프로세스가 선점(preempted) 되는가?
  - 프로세스가 받은 실행 시간은 얼마나 되는가?
  - 프로세스가 완전히 처리되는 데 필요한 시간은 얼마나 더 요구되는가?

스케줄링의 분류(1/2)

❖ **프로세스 상태 변화와 실행 빈도에 따라 스케줄링 분류**

-장기 스케줄러 : 스케줄링에 따라 디스크에서 메모리로 작업 가져와 처리할 순서 결정과 메모리의 사용 가능 공간 확인.

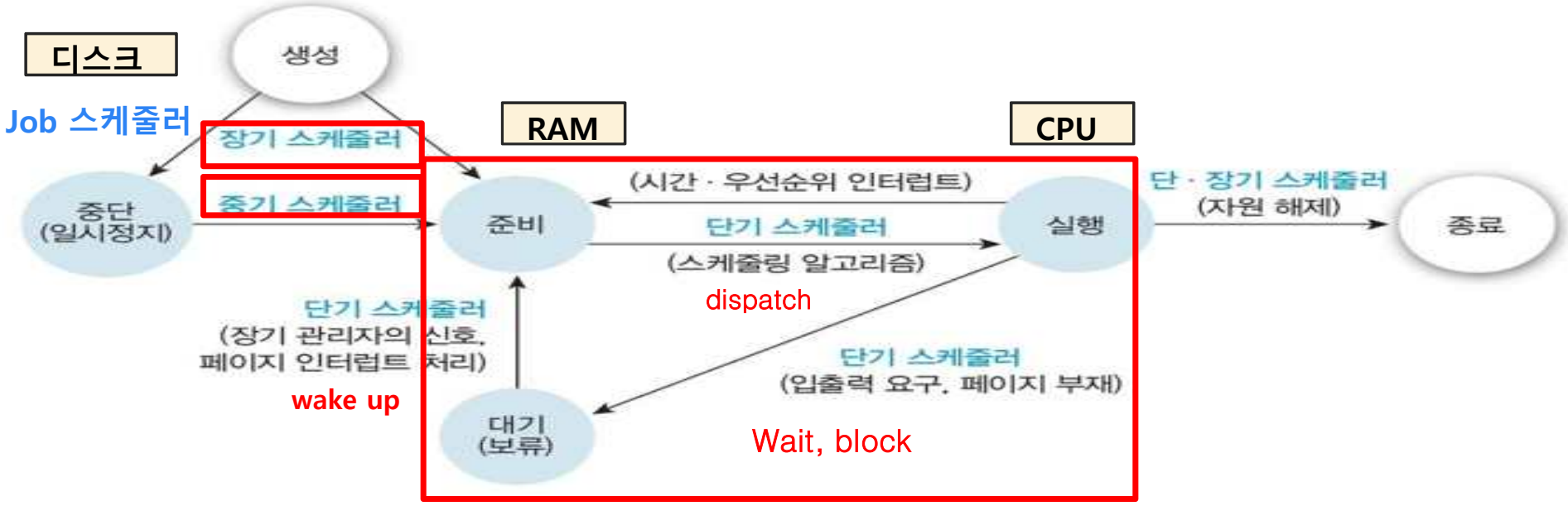
실행 빈도가 적어 드물게 수행 (작업이 생성 될 때와 종료 될 때 실행됨)

일괄처리 시스템은 한꺼번에 요청한 작업이 디스크에 존재하므로 장기 스케줄러가 작업을 처리할 순서 결정

-중기 스케줄러: 시스템의 오버헤드에 따라 연기할 프로세스를 잠정적으로 결정하여 메모리에서 제거하여 디스크에 저장

실행 빈도가 중간으로 메모리 사용성을 높이고 작업 효율성 향상을 위함 (시분할 시스템은 장기 스케줄러는 없고 중기 스케줄러를 사용)

-단기 스케줄러 : 메모리에 적재된 프로세스 중 프로세서를 할당하여 실행 상태가 되도록 결정, 실행 빈도가 많고 처리속도가 매우 빠름



스케줄링의 분류(2/2)

- 스케줄링(Scheduling)의 분류
- 장기 스케줄링(작업 스케줄링)
  - 스케줄링에 따라 디스크에서 메모리로 작업 가져와 처리할 순서 결정. 제출 시간, 작업 이름, 작업 길이(용량) 등의 정보 필요
  - 어떤 작업에게 시스템의 자원들을 차지할 수 있도록 할 것인가를 결정하여 준비상태 큐로 보내는 작업을 의미
  - 작업(job) 스케줄링, 장기 스케줄링이라고도 하며 작업 스케줄러에 의해 수행된다.
- 중기 스케줄링
  - 어떤 프로세스들이 CPU를 할당 받을 것인지 결정하는 작업을 의미
  - CPU를 할당 받으려는 프로세스가 많은 경우 프로세스를 일시 보류 시킨 후 활성화해서 일시적으로 부하를 조절한다.(짧은 순간에 프로세스들에 대한 일시적인 활동의 중단 및 재개를 수행) 또한 중기 스케줄링이라고도 한다.
- 단기 스케줄링(프로세서 스케줄링)
  - 메모리에 적재된 프로세스 중 프로세서를 할당하여 실행 상태가 되도록 결정하는 프로세스 스케줄링
  - 어떤 준비완료 프로세스(ready process)에게 중앙처리장치를 할당할 것인가를 결정
  - 프로세스가 실행 되기 위해 CPU를 할당 받는 시기와 특정 프로세스를 지정하는 작업
  - 프로세서 스케줄링, 단기 스케줄링이라고도한다.
  - 프로세서 스케줄링 및 문맥 교환은 프로세서 스케줄러에 의해 수행된다.
  - 준비 상태의 프로세스에 프로세서 할당(디스패칭)

스케줄링의 문맥 교환과 디스패처

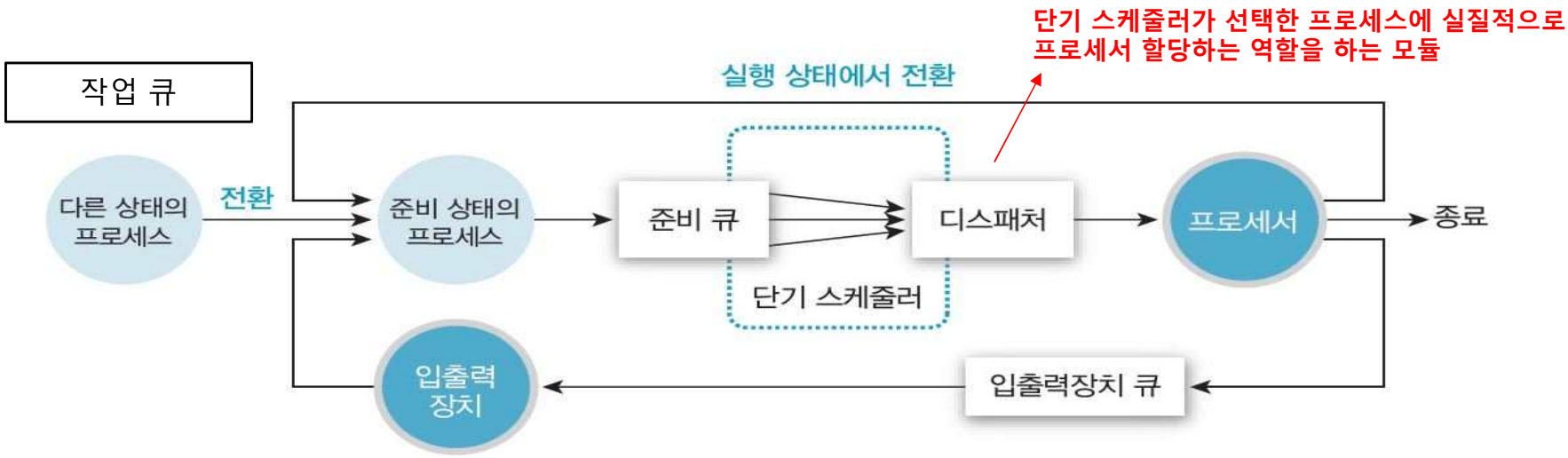
- 문맥교환(Context Switching)
  - 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것으로, 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장하고, 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 하는 작업, **프로세서(cpu) 스케줄러의 디스패처에 의해 수행**

- 디스패처(Dispatcher)
  - CPU 스케줄러 내부에 포함된 것으로, 단기 스케줄러가 선택한 프로세스에 실질적으로 프로세서를 할당하는 역할을 한다.
  - 프로세스의 레지스터를 적재하고(문맥교환) 프로세스가 다시 시작할 때 사용자 프로그램이 올바른 위치를 찾을 수 있도록 한다.

스케줄링하기 위한 큐 종류

- 스케줄링 개념
  - 스케줄링은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미한다.
  - 프로세스가 생성되어 완료될 때까지 프로세스는 여러 종류의 스케줄링 과정을 거치게 된다.

  - 스케줄링 하기 위한 큐(Queue)
    - 작업 큐(Job Queue), 준비 큐(Ready Queue), 장치 큐(Device Queue)



스케줄링의 성능 기준

- 스케줄링의 목적 중 CPU이용률, 처리율, 반환시간, 대기 시간, 응답 시간은 여러 종류의 스케줄링 성능을 비교하는 기준이 된다.
  - 처리율(Throughput) 증가 : 단위 시간당 프로세스를 처리하는 비율을 증가시킨다.
  - CPU이용률 (CPU Utilization) 증가 : 프로세스 실행과정에서 주기억장치를 액세스 한다든지 입/출력 실행등의 원인에 의해 발생할 수 있는 CPU낭비시간을 줄이고, CPU가 순수하게 프로세스를 실행하는데 사용되는 시간 비율을 증가시킨다.
  - 응답시간(Response Time) 최소화 : 작업을 지시하고, 반응하기 시작하는 시간을 최소화한다.
  - 반환시간(Turn around time) 최소화 : 프로세스를 제출한 시간부터 실행이 완료될때까지 걸리는 시간을 최소화한다.
  - 대기시간(Waiting Time) 최소화 : 프로세스가 중비상태 큐에서 대기하는 시간을 최소화한다.

스케줄링의 성능 기준

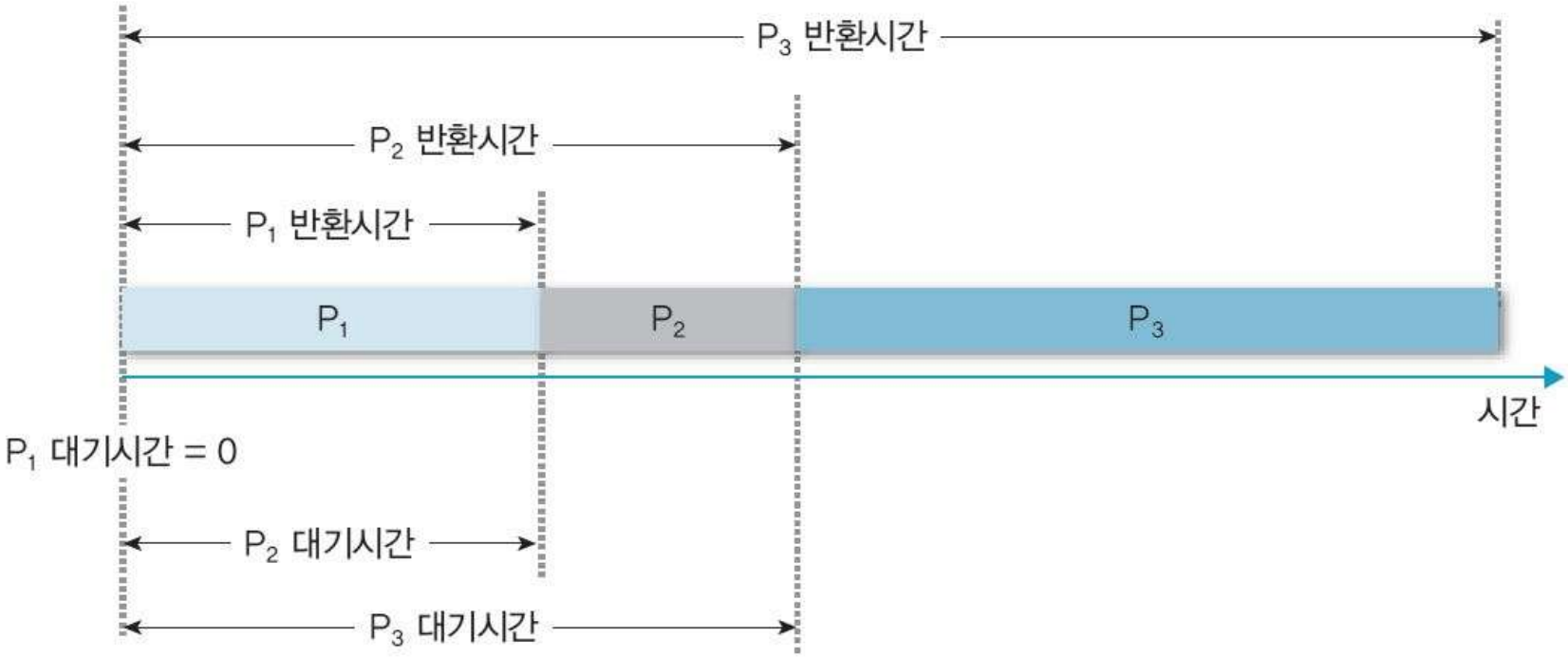
- 스케줄링의 목적 중 CPU이용률, 처리율, 반환시간, 대기 시간, 응답 시간은 여러 종류의 스케줄링 성능을 비교하는 기준이 된다.
  - **처리율(Throughput) 증가,CPU이용률 (CPU Utilization) 증가**
  - **응답시간(Response Time),반환시간(Turn around time) ,대기시간(Waiting Time) 최소화**

반환시간, 대기시간, 응답시간의 관계

대기시간= 준비 큐에 머무는 시간
응답시간=작업을 요청한 시간부터 반응을 시작하는 시간까지 간격(첫번째 응답) 대화식 작업에서 중요
반환시간=작업이 메모리에 들어가기까지 걸린 시간 +준비 큐에 머무는 시간 + 실행 시간+입출력 시간

스케줄링의 성능 기준

- **프로세스의 반환 시간과 대기 시간**



프로세스 P_1, P_2, P_3의 반환시간, 대기시간 프로세스 3개가 동시에 도착하여 P1, P2, P3 순으로 실행

프로세스 스케줄링의 기법

- **방법, 환경별 분류**

❖ 선점/비 선점(preemptive/nonpreemptive) 스케줄링

- 비 선점스케줄링
  - 하나의 프로세스에 중앙처리장치가 할당되면 그 프로세스의 수행이 끝날 때까지 중앙처리장치는 그 프로세스로부터 빠져나올 수 없다. 일괄 처리 방식에 적합하다.
- 선점스케줄링
  - 하나의 프로세스가 중앙처리장치를 차지하고 있을 때 다른 프로세스가 현재 수행 중인 프로세스를 중지시키고 자신이 중앙처리장치를 차지할 수 있다. 시분할 시스템, 온라인 응용들에 적합하다.

❖ 우선순위(priority) 스케줄링

- 각 프로세스에게 우선순위를 부여하여 우선순위가 높은 순서대로 처리하는 방법
- 정적 우선순위(static priority) 기법
  - 상대적으로 오버헤드는 적으나, 주위 여건의 변화에 적응하지 않고 우선순위를 바꾸지 않는다.
- 동적 우선순위(dynamic priority) 기법
  - 필요에 따라 우선순위 재구성

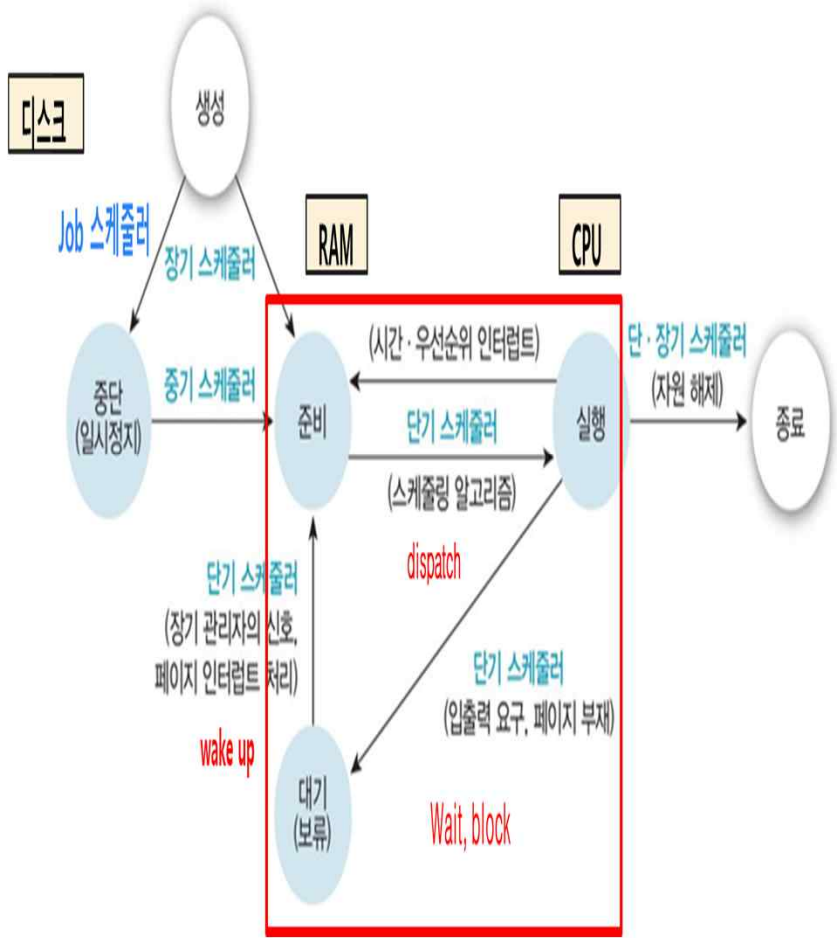
❖ 기한부(deadline) 스케줄링

- 작업들이 명시된 시간이나 기한 내에 완료되도록 계획
- 정적(static)스케줄링 방식, 동적(dynamic)스케줄링 방식

프로세스 스케줄링의 기법

❖ **선점/비 선점(preemptive/nonpreemptive) 스케줄링**

- 비 선점스케줄링
  - 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법(일괄 처리 방식에 적합)
- 선점스케줄링
  - 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 있는 스케줄링 기법. (시분할 시스템에 적합)



❖ **스케줄링이 필요한 경우**

1.Running -> Blocked (ex: I/O 요청에 의한 call) => 비선점

2.Running -> Ready (ex: timer interrupt) => 선점

3.Blocked -> Ready (ex: I/O 완료에 의한 interrupt) => 선점

4.Terminate => 비선점

◆ **비선점 : 자진해서 CPU를 반납하고 문맥교환**

선점 : CPU를 빼앗기고 문맥교환

비선점 스케줄링

- 비선점 스케줄링
  - FCFS(First Come First Service)
  - SJF(Shortest Job First)
  - HRN (Highest Response-ratio Next)
  - 기한부(Deadline)
  - 우선순위(Priority)

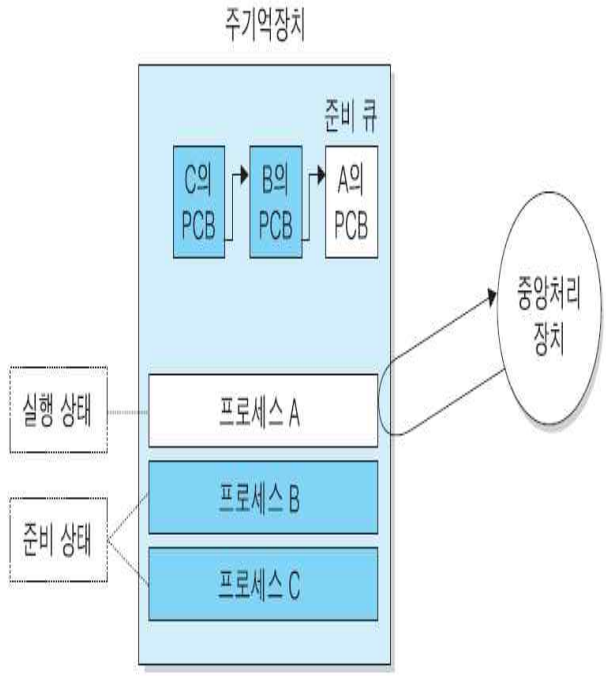
비선점 기법(nonPreemptive)

- 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법이다.
- 일괄 처리 방식에 적합하다.
- 모든 프로세스에 대한 요구를 공정하게 처리할 수 있다.
- FCFS, SJF, 우선순위, HRN, 기한부 등의 알고리즘이 있다.

Cf)선점스케줄링 : RR, SRT, 선점우선순위, 다단계큐, 다단계피브맥 큐

(1) FCFS(First Come First Service)

- 선입선출, FIFO(First in First Out) 이다.
- FCFS는 준비 상태 큐(대기 큐, 준비완료 리스트, 작업준비큐, 스케줄링큐)에 도착한 순서에 따라 차례로 CPU를 할당하는 기법으로 , 가장 간단한 알고리즘이다.
- 문제점
  - 먼저 도착한 것이 먼저 처리되어 공평성은 유지되지만 짧은 작업이 긴 작업을 , 중요한 작업이 중요하지 않은 작업을 기다리게 한다.



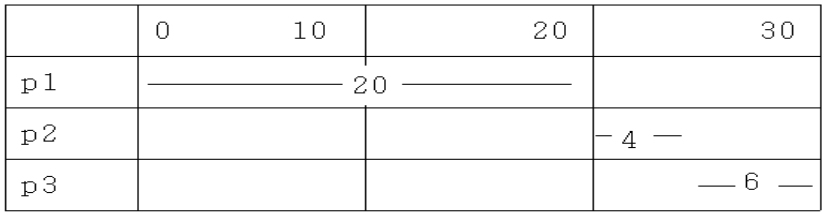
FCFS(First Come First Service) 예제

❖ 다음과 같은 프로세스들이 차례로 준비상태큐에 들어왔다고 가정할 때 FCFS기법을 이용하여 평균 실행시간, 평균 대기시간, 평균 반환시간을 작성하시오

프로세스번호	p1	p2	p3
실행시간(초)	20	4	6

❖ 대기시간 : 프로세스의 대기한 시간으로 , 바로 앞 프로세스까지의 진행시간을 구함

❖ 반환시간 : 프로세스의 대기시간과 실행 시간의 합



- 평균 실행시간 : (20+4+6)/3 = 10
- 평균 대기시간 : (0+20+24)/3=14.6
- 평균 반환시간 : (20+24+30)/3=24.6

FCFS(First Come First Service) 문제

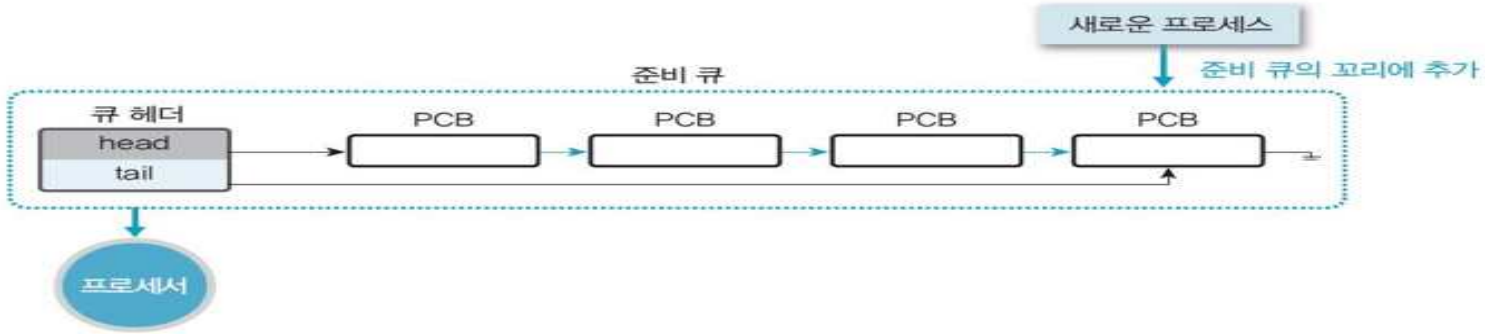
❖ FIFO기법을 적용하여 작업스케줄링을 하였을 때 , 다음 작업들의 평균 회수시간(turn around time)은?(단 문맥교환 시간은 무시한다.)

가.6.75 나.7.25 다.7.75 라,8.25

작업	도착시간	실행시간
A	0	6
B	1	3
C	2	1
D	3	4

(1) FCFS(First Come First Service)

- 선입선출, FIFO(First in First Out) 이다.
- FCFS는 준비 상태 큐(대기 큐, 준비완료 리스트, 작업준비큐, 스케줄링큐)에 도착한 순서에 따라 차례로 CPU를 할당하는 기법으로 , 가장 간단한 알고리즘이다.
- 문제점
  - 먼저 도착한 것이 먼저 처리되어 공평성은 유지되지만 짧은 작업이 긴 작업을 , 중요한 작업이 중요하지 않은 작업을 기다리게 한다.
- 선입선출 스케줄링 알고리즘
  - 새로운 프로세스가 들어오면 해당 프로세스의 PCB(프로세스 제어 블록)을 준비 큐의 마지막에 연결한다.그리고 차례가 되면 준비 큐의 앞 부분에 있던 프로세스가 프로세서를 할당 받고 준비 큐에서 삭제된다.



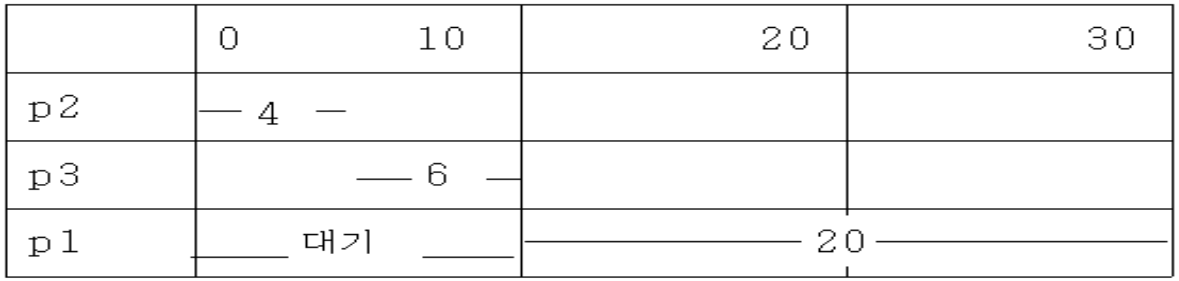
(2) SJF(Shortest Job First)

- 준비상태큐에서 기다리고 있는 프로세스들 중에서 실행시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법이다
- 가장 적은 평균 대기시간을 제공하는 최적 알고리즘이다.
- 문제점
  - 실행시간이 긴 프로세스는 실행시간이 짧은 프로세스에게 할당 순위가 밀려 무한 연기 상태가 발생할 수 있다.

SJF(Shortest Job First) 예제

- 다음과 같은 프로세스들이 차례로 준비상태큐에 들어왔다고 가정할 때 SJF기법을 이용하여 평균 실행시간, 평균 대기시간, 평균 반환시간을 작성하시오.(도착 시간이 없는 경우)

프로세스번호	p1	p2	p3
실행시간(초)	20	4	6



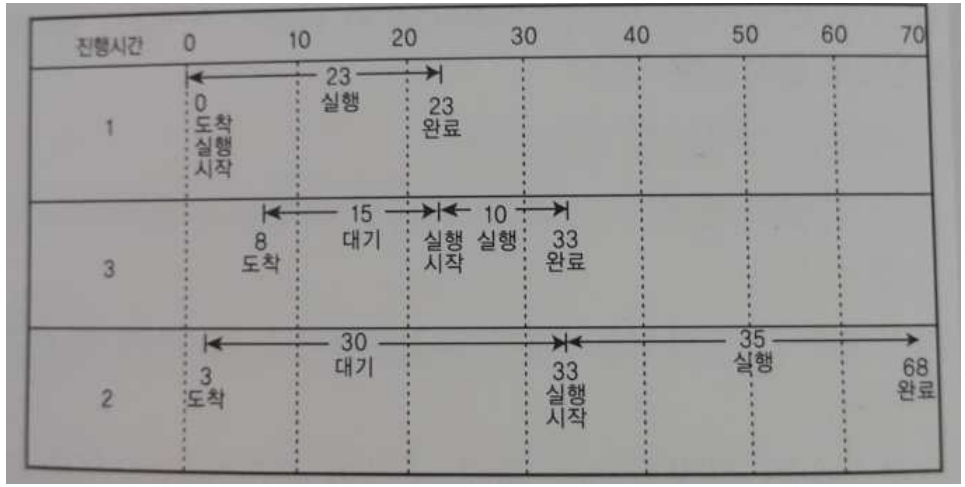
- 도착 시간이 없는 경우는 실행 시간이 짧은 것부터 먼저 처리한다.
- 평균 실행시간 : (4+6+20)/3 = 10
- 평균 대기시간 : (0+4+10)/3=4.6
- (10초까지 p1은 대기 11초부터 30초까지 20초 실행)
- 평균 반환시간 : (4+10+30)/3=14.6

SJF(Shortest Job First) 문제

❖ SJF(Shortest Job First)스케줄링에서 작업 도착 시간과 CPU사용시간은 아래표와 같다. 모든 작업들의 평균 대기 시간은 얼마인가? (**도착 시간이 있는 경우**)

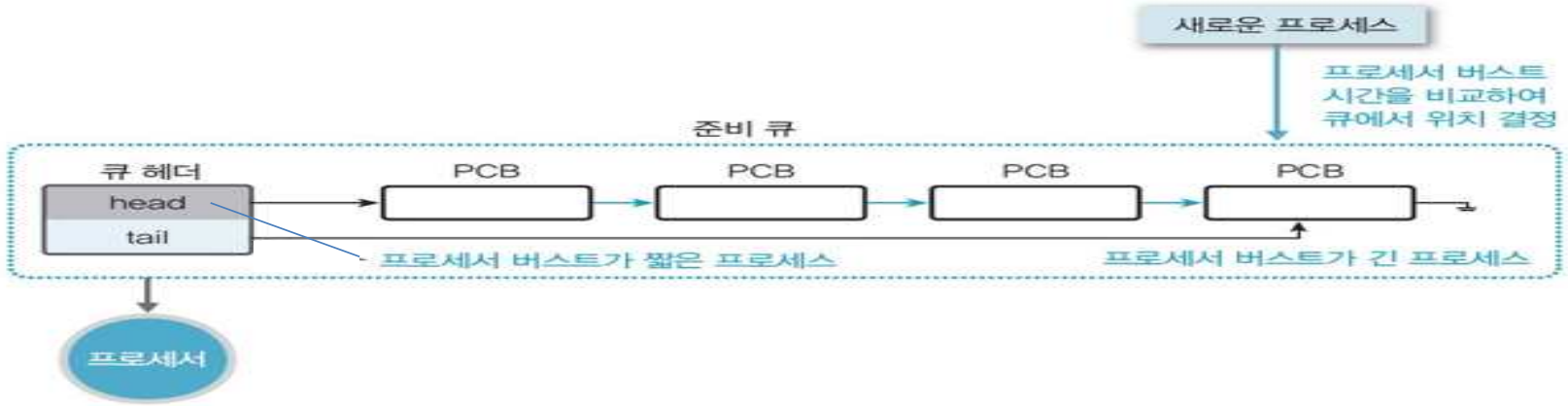
가. 15　　나. 17　　다. 24　　라. 25

작업	도착시간	CPU사용시간
1	0	23
2	3	35
3	8	10



(2) SJF(Shortest Job First)

- 준비상태큐에서 기다리고 있는 프로세스들 중에서 실행시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법이다
- 가장 적은 평균 대기시간을 제공하는 최적 알고리즘이다.
- 문제점
  - 실행시간이 긴 프로세스는 실행시간이 짧은 프로세스에게 할당 순위가 밀려 무한 연기 상태가 발생할 수 있다.
- 최소 작업 우선 스케줄링 알고리즘
  - 새로운 프로세스가 들어오면 준비 큐에 들어 오면 각 프로세스의 실행 시간과 비교하여 위치를 결정한다. 준비 큐의 앞 부분에 있던 프로세스가 프로세서를 할당 받고 준비 큐에서 삭제된다.



(3) HRN (Highest Response-ratio Next)

❖ SJF 알고리즘의 단점을 보완한 기법

=>CPU실행시간이 짧은 작업에 우선권을 주기 때문에 실행시간이 긴 프로세스는 대기 시간(waiting time)이 길어져 불리하다는 SJF기법 단점을 보완한 기법

❖ SJF 알고리즘과 마찬가지로 비선점 방식 수행

❖ 우선 순위 결정하여 우선 순위가 높은 프로세스를 처리하는 기법

=>우선순위를 계산하여 그 숫자가 가장 높은 것부터 낮은 순으로 우선순위가 부여

❖ 각 작업에 대한 처리의 우선순위를 결정 방법

=>CPU실행(서비스)시간과 대기 시간을 이용

=>해당 작업이 요구하는 CPU 실행 시간 이외에 대기하고 있는 시간까지도 고려하여 선정

(3) HRN (Highest Response-ratio Next)

❖ 우선 순위 결정하여 우선 순위가 높은 프로세스를 처리하는 기법

❖ 각 작업에 대한 처리의 우선순위를 결정 방법

=> CPU실행(서비스)시간과 대기 시간을 이용

❖ 우선순위 계산식(시스템 응답시간)

- 처리 우선순위 값 = $\frac{(\text{대기시간}+\text{CPU실행(서비스)시간})}{\text{CPU실행(서비스)시간}}$

=>이 식의 결과값을 응답 비율(response ratio)이라고 하며, 1일 때 최소값

=>식의 결과가 1 이라는 것은 CPU실행시간의 크기에 관계없이 대기한 시간이 없기 때문에 그 결과는 1 이 됨(제일 처음으로 시스템에 들어온 작업에 해당)

=>대기 시간이 있는 작업 간에는 식의 결과값이 큰 프로세스에 대해서 높은 우선 순위를 부여함

(3) HRN (Highest Response-ratio Next)

❖ 우선순위 계산식(시스템 응답시간)

- 처리 우선순위 값 = $\frac{\text{(대기시간+CPU실행(서비스)시간)}}{\text{CPU실행(서비스)시간}}$

=>분모에 CPU 실행시간이 있으므로, 똑같은 대기 시간을 갖는 작업간에는 CPU실행 시간이 짧은 것이 우선 순위가 높게 된다.

=>분자에 대기 시간이 있기 때문에 똑같은 CPU 실행 시간을 가진 작업 사이에는 대기 시간이 긴 것에 높은 우선 순위를 부여된다.

=>서비스 실행시간이 짧거나 대기시간이 긴 프로세스일 경우 우선순위가 높아진다.

=>우선순위 계산 공식을 이용하여 서비스시간이 짧은 프로세스나 대기시간이 긴 프로세스에게 우선순위를 주어 CPU를 할당한다.

=>시스템 응답시간 값이 클수록 우선순위가 높아진다.

HRN (Highest Response-ratio Next) 예제

❖ 다음과 같은 프로세스가 HRN 기법으로 스케줄링될 때 우선순위를 계산하시오.

프로세스번호	p1	p2	p3
실행시간(초)	20	4	6
대기시간	10	20	10

우선순위 계산	(10+20)/20=1.5	(20+4)/4=6	(10+6)/6=2.6
우선순위	P2 -> p3 -> p1		

❖ 우선순위 계산식(시스템 응답시간)

처리 우선순위 값 = (대기시간+CPU실행(서비스)시간) / CPU실행(서비스)시간

❖ 분자에 대기 시간이 있기 때문에 똑같은 CPU 실행 시간을 가진 작업 사이에는 대기 시간이 긴 것에 높은 우선 순위를 부여된다.

❖ 서비스 실행시간이 짧거나 대기시간이 긴 프로세스일 경우 우선순위가 높아진다.

❖ 우선순위를 계산하여 그 숫자가 가장 높은 것부터 낮은 순으로 우선순위가 부여된다.

❖ P2가 가장 응답시간이 높으므로 , P3, P1의 우선 순위에 따라 cpu를 할당 받는다.

HRN (Highest Response-ratio Next) 문제

❖ 아래의 작업 중 운영체제가 CPU스케줄링 기법으로 HRN방식을 구현했을 때 우선순위가 가장 높은 것은?

가.a 나.b 다.c 라.d

작업번호	a	b	c	d
대기시간	20	**5**	3	**5**
서비스시간	5	5	12	3

❖ 우선순위 계산식(시스템 응답시간)

처리 우선순위 값 = $\frac{(\text{대기시간}+\text{CPU실행(서비스)시간})}{\text{CPU실행(서비스)시간}}$

(4) 기한부(Deadline)

- 프로세스에게 일정한 시간을 주어 그 시간 안에 프로세스를 완료하도록 하는 기법
- 작업들의 결과가 시간 내에 구해지면 매우 효율성이 높다
- 프로세스가 제한된 시간 안에 완료되지 않을 경우 제거되거나 처음부터 다시 실행해야 한다.
- 시스템은 프로세스에게 할당할 정확한 시간을 추정해야 하며, 이를 위해서 사용자는 시스템이 요구한 프로세스에 대해 정확한 정보를 제공해야 한다=>정확한 정보를 미리 예측하기 어려움

- 문제점
  - 여러 프로세스들이 동시에 실행되면 스케줄링 복잡해지며, 프로세스 실행 시 집중적으로 요구되는 자원관리에 오버헤드가 발생한다.

(5) 우선순위(Priority)

- 준비상태 큐에서 기다리는 각 프로세스마다 우선순위(여러가지 요인으로 등급 부여)를 부여하여 그 중 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법이다.
- 우선순위가 동일할 경우 FCFS기법으로 CPU를 할당한다.
- 우선순위는 프로세스의 종류나 특성에 따라 다르게 부여된다.
- 가장 낮은 순위를 부여받은 프로세스는 무한연기 또는 기아상태(Starvation)가 발생할 수 있다.

=>무한연기 또는 기아상태(Starvation) 해결 방안

[Aging기법]

- 시스템에서 특정 프로세스의 우선순위가 낮아 무한정 기다리게 되는 경우 , 한번 양보하거나 기다린 시간에 비례하여 일정시간이 지나면 우선순위를 한 단계씩 높여 가까운 시간안에 자원을 할당받도록 하는 기법이다.
- SJF나 우선순위 기법에서 발생할 수 있는 무한 연기 상태, 기아상태를 예방할 수 있다.

(5) 우선순위(Priority) 문제

❖ 프로세스가 자원을 기다리고 있는 시간에 비례하여 우선순위를 부여함으로써 무기한 문제를 방지하는 기법은?

가. 노화(aging)기법

나. 재사용(Reusable) 기법

다. 환형대기(Circular)기법

라. 치명적인 포옹(Deadly Embrace)

학습 내용 정리

비선점 기법(nonPreemptive) 스케줄링 기법

FCFS(First Come First Service)	• 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법 • 먼저 도착한 것이 먼저 처리되어 공평성은 유지되지만 짧은 작업이 긴 작업을, 중요한 작업이 중요하지 않은 작업을 기다리게 됨
SJF(Shortest Job First)	• 실행 시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법 • 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘
HRN(Hightest Response-ratio Next)	• 실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법 • 우선순위 계산 공식 = $\frac{\text{대기 시간} + \text{서비스 시간}}{\text{서비스 시간}}$
기한부(Deadline)	• 프로세스에게 일정한 시간을 주어 그 시간 안에 프로세스를 완료하도록 하는 기법 • 시스템은 프로세스에게 할당할 정확한 시간을 추정해야 하며, 이를 위해서 사용자는 시스템이 요구한 프로세스에 대한 정확한 정보를 제공해야 함
우선순위 (Priority)	• 기다리는 각 프로세스마다 우선순위를 부여하여 그 중 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법 • 우선순위의 등급은 내부적 요인과 외부적 요인에 따라 부여할 수 있음 • 각 작업마다 우선순위가 주어지며, 우선순위가 제일 높은 작업에게 먼저 프로세서가 할당됨 • 가장 낮은 순위를 부여받은 프로세스는 무한 연기 또는 기아 상태(Starvation)가 발생할 수 있음