Chatpter 06
프로세스 스케줄링

그림으로 배우는 구조와 원리

운영체제 개정3판

01 스케줄링의 이해

02 스케줄링 알고리즘

03 스케줄링 알고리즘의 평가

- 스케줄링 단계를 이해한다.
- 장기 스케줄러와 단기 스케줄러의 역할을 알아본다.
- 여러 프로세스 스케줄링 알고리즘의 특성을 이해한다.
- 알고리즘의 성능을 알아본다.

Section 01 스케줄링의 이해(1. 스케줄링의 개념)

- **스케줄링scheduling의 개념**
  - 여러 프로세스가 번갈아 사용하는 자원을 어떤 시점에 어떤 프로세스에 할당할지 결정
  - 자원이 프로세서인 경우를 프로세서 스케줄링, 대부분의 스케줄링이 프로세서 스케줄링 의미
  - 스케줄링 방법에 따라 프로세서를 할당받을 프로세스 결정하므로 스케줄링이 시스템의 성능에 영향 미침
  - 좋은 스케줄링은 프로세서 효율성 높이고, 작업(프로세스)의 응답시간 최소화하여 시스템의 작업 처리 능력 향상
- **스케줄링의 목적**
  - **자원 할당의 공정성** 보장
  - 단위시간당 **처리량 최대화**
  - **적절한 반환시간 보장**
  - **예측 가능성 보장**
  - **오버헤드 최소화**

3. 스케줄링의 기준 요소

- **프로세스의 실행**

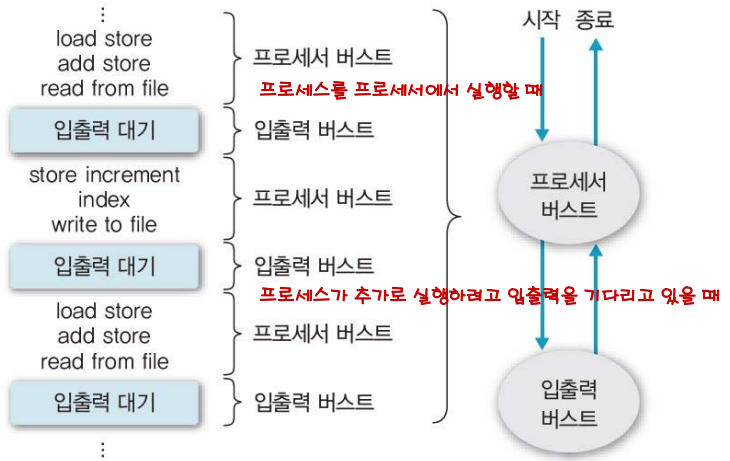


그림 6-1 실행(프로세서 버스트)과 입출력 대기(입출력 버스트)의 순환

3. 스케줄링의 기준 요소

- 프로세스 버스트 시간 그래프

그림 6-2 프로세서 버스트 시간 그래프

3. 스케줄링의 기준 요소

- 프로세서 버스트가 긴 예와 짧은 예

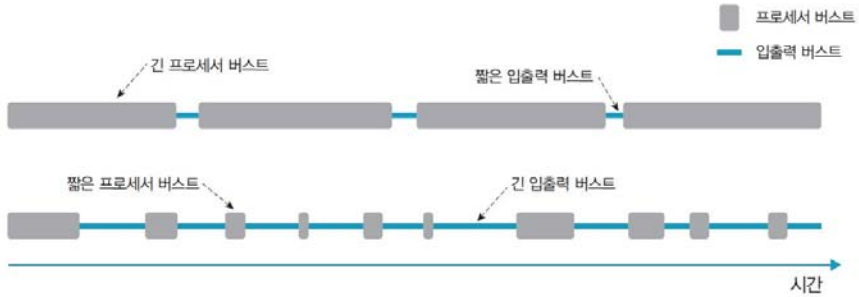


그림 6-3 프로세서 버스트가 긴 예와 짧은 예

- **CPU 버스트 프로세스** : 수학 연산과 같이 CPU를 많이 사용하는 프로세스로 CPU 버스트가 많은 프로세스
- **입출력 버스트 프로세스** : 저장장치에서 데이터를 복사하는 일과 같이 입출력을 많이 사용하는 프로세스로 입출력 버스트가 많은 프로세스

5. 스케줄링 큐

- 스케줄링 큐

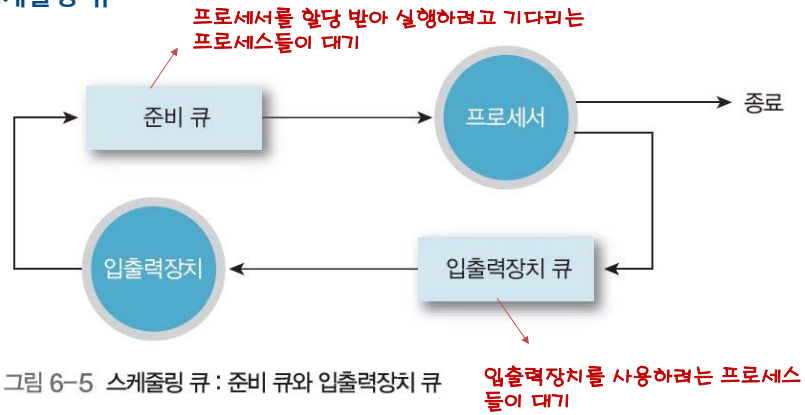


그림 6-5 스케줄링 큐 : 준비 큐와 입출력장치 큐

5. 스케줄링 큐

- 다단계 큐

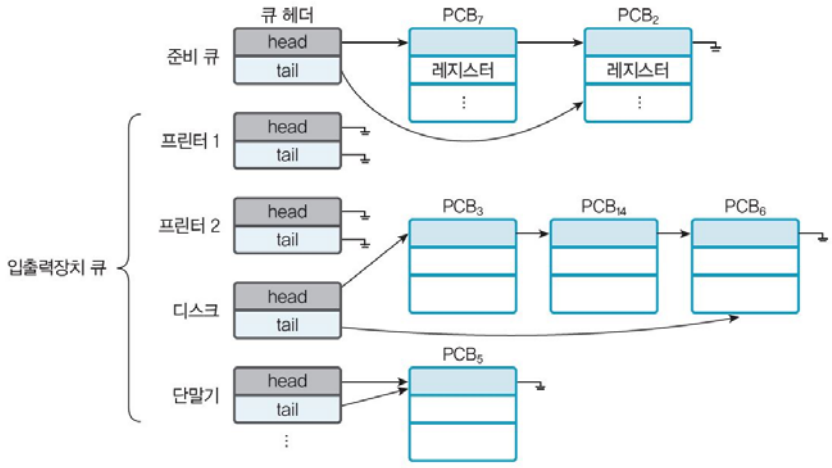


그림 6-6 다단계 큐 예 : 준비 큐와 다양한 입출력장치 큐

6. 스케줄링과 스케줄러

- **큐잉queueing diagram 도표**
  - 프로세스 스케줄링 표현하는 방법
  - 예

❶ 프로세스가 입출력 요청 보내고 입출력 큐에 들어감
❷ 프로세스가 새로운 프로세스를 생성fork하고 생성한 프로세스의 종료 대기
❸ 프로세스가 시간 할당량을 초과(시간 종료)하면 준비 큐에 들어감
❹ 인터럽트로 프로세서에서 제거된 프로세스는 다시 준비 큐에 들어감

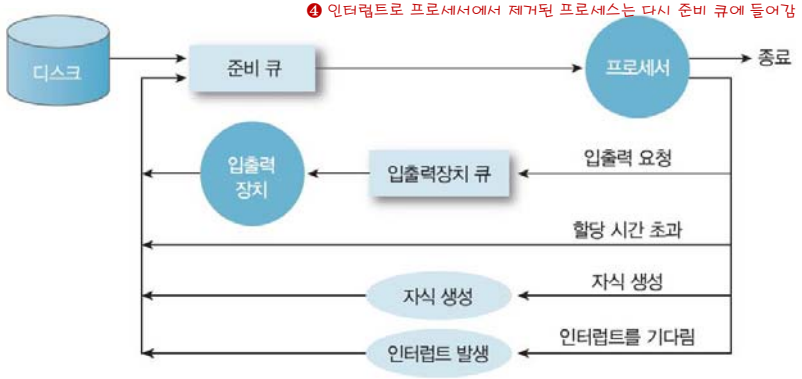


그림 6-7 큐잉 도표 예

❶, ❷의 경우 프로세스는 대기 상태에서 준비 상태로 전환, 다시 준비 큐에 들어감. 프로세스는 종료할 때까지, 이 순환 반복

6. 스케줄링과 스케줄러

- **스케줄러의 종류와 역할**

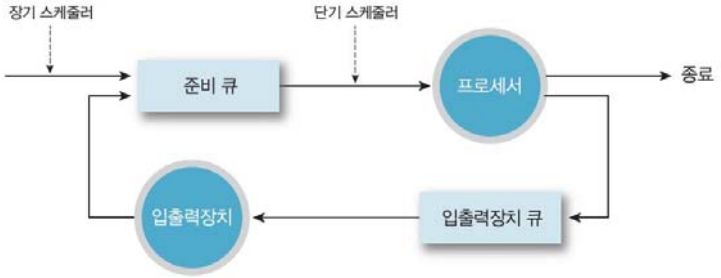


그림 6-8 스케줄러 설명에 사용할 단순 큐잉 도표 예

- 장기 스케줄러
  - 작업 스케줄러라고도 하며, 스케줄링에 따라 디스크에서 메모리로 작업 가져와 처리할 순서 결정. 제출 시간, 작업 이름, 작업 길이(용량) 등의 정보 필요
- 단기 스케줄러
  - 메모리에 적재된 프로세스 중 프로세서를 할당하여 실행 상태가 되도록 결정하는 프로세스 스케줄링을 한다. 이때는 프로세스가 실행하는 데 필요한 자원의 요청 만족해야 함

6. 스케줄링과 스케줄러

- 장기 스케줄러와 단기 스케줄러 차이

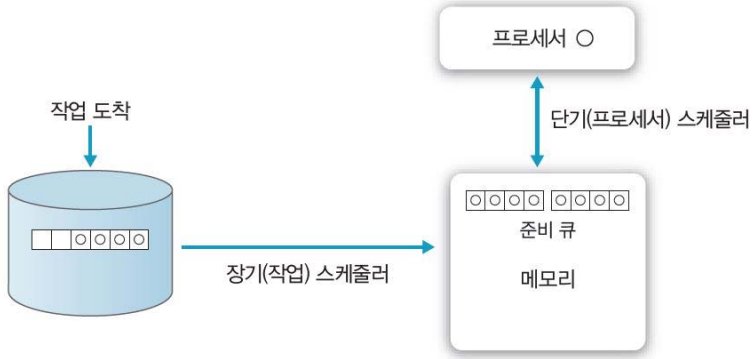


그림 6-9 장기(작업) 스케줄러와 단기(프로세서) 스케줄러

장기 스케줄러와 단기 스케줄러의 가장 큰 차이는 실행 빈도
단기 스케줄러는 실행할 프로세스 수시로 선택, 매우 빨라야 함
장기 스케줄러는 시스템에 새로운 작업이 분minute 단위로 들어오므로 단기 스케줄러에 비해 상대적으로 드물게 수행. 장기 스케줄러는 다중 프로그래밍의 정도degree of multiprogramming(메모리에 있는 프로세스 수) 결정

7. 선점 스케줄링과 비선점 스케줄링

- **선점 스케줄링(Preemptive Scheduling)**
  - **프로세스 하나가 장시간 동안 프로세서를 독점하는 것을 방지**하여 모든 프로세스에 프로세서를 서비스할 기회 늘림
  - **운영체제가 필요하다고 판단하면 실행 상태에 있는 프로세스의 작업을 중단시키고 새로운 작업을 시작할 수 있는 방식**
  - 우선순위가 높은 프로세스들이 긴급 처리 요청할 때 유용
  - 대화식 시분할 시스템이나 실시간 시스템에서 빠른 응답시간 유지 위해 필수
  - 오버헤드가 커질 수 있어 효과적인 이용을 위해서는 메모리에 프로세스가 많은 적재 필요

- **비선점 스케줄링(Non-preemptive Scheduling)**
  - **어떤 프로세스가 실행 상태에 들어가 CPU를 사용하면 그 프로세스가 종료되거나 자발적으로 대기 상태에 들어가기 전까지는 계속 실행되는 방식**
  - 실행 시간이 짧은 프로세스(작업)가 실행 시간이 긴 프로세스(작업)를 기다리는 대신 모든 프로세서 공정 관리
  - 우선순위가 높은 프로세스 중간에 입력해도 대기 중인 프로세스는 영향을 받지 않으므로 응답시간 예측 용이

8. 스케줄링 알고리즘의 선점 기준

- 스케줄링 알고리즘의 선택 기준(평가 기준)
  - 프로세서 사용률 (CPU Utilization)
  - 처리율(Throughput)
  - 반환시간(Turnaround Time)
  - 대기시간(Waiting Time)
  - 반응시간(Response Time)

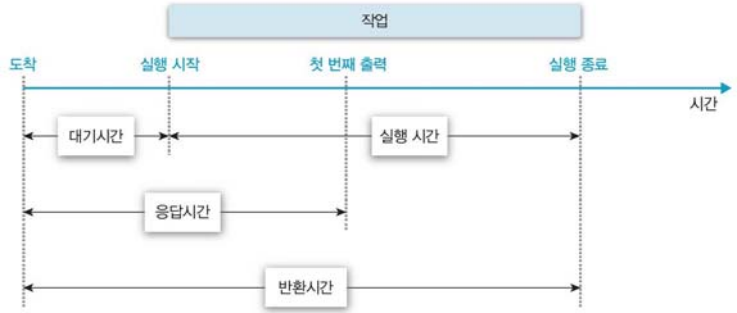
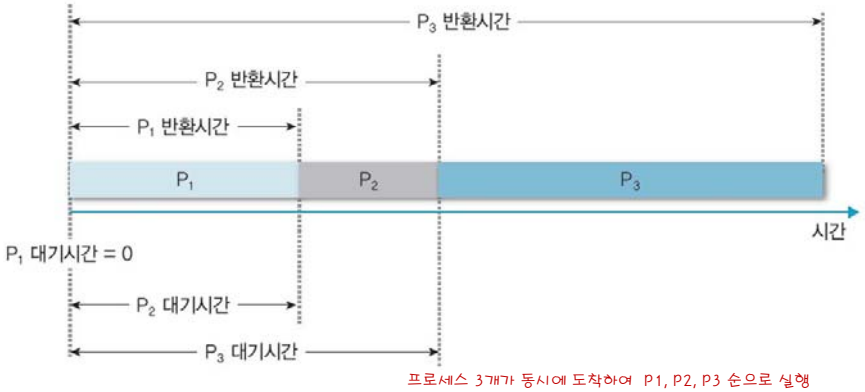


그림 6-13 반환시간, 대기시간, 응답시간의 관계

8. 스케줄링 알고리즘의 선점 기준

- 프로세스의 반환시간과 대기시간



프로세스 3개가 동시에 도착하여 P1, P2, P3 순으로 실행

그림 6-14 프로세스 P_1, P_2, P_3의 반환시간, 대기시간

Section 02 스케줄링 알고리즘(1. 선입선처리 스케줄링)

- 선입선처리FCFS, First Come First Served 스케줄링의 개념
  - 비선점 방법으로 프로세서 스케줄링 알고리즘 중 가장 단순
  - **프로세서 요청하는 순서대로 프로세서 할당, 선입선출FIFO 큐로 구현**
  - 일괄 처리 시스템에서는 매우 효율적이나 빠른 응답을 요청하는 대화식 시스템에는 적합하지 않음
  - 원리

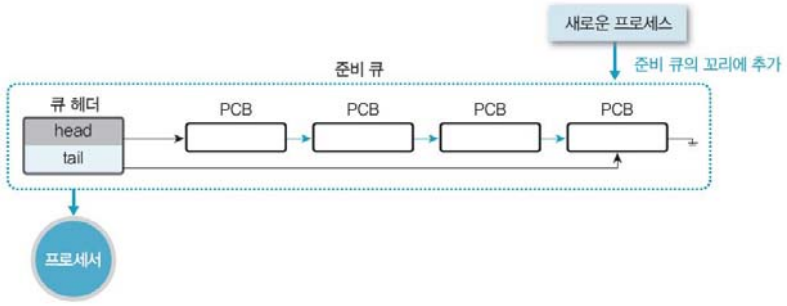


그림 6-15 선입선처리 스케줄링

1. 선입선처리 스케줄링

- 선입선처리 스케줄링의 예 1

프로세스	도착 시간	실행 시간
P_1	0	10
P_2	1	28
P_3	2	6
P_4	3	4
P_5	4	14

(a) 준비 큐

(b) 간트 차트

표 6-1 그림 6-16 예의 반환시간과 대기시간

프로세스	반환시간	대기시간
P_1	10	0
P_2	(38 − 1) = 37	(10 − 1) = 9
P_3	(44 − 2) = 42	(38 − 2) = 36
P_4	(48 − 3) = 45	(44 − 3) = 41
P_5	(62 − 4) = 58	(48 − 4) = 44
평균 반환시간 : 38.4[= (10 + 37 + 42 + 45 + 58)/5]		평균 대기시간 : 26[= (0 + 9 + 36 + 41 + 44)/5]

1. 선입선처리 스케줄링

- 호위효과(Convoy Effect)

프로세서 중심 프로세스 하나가 CPU를 떠나기를 기다리는 현상

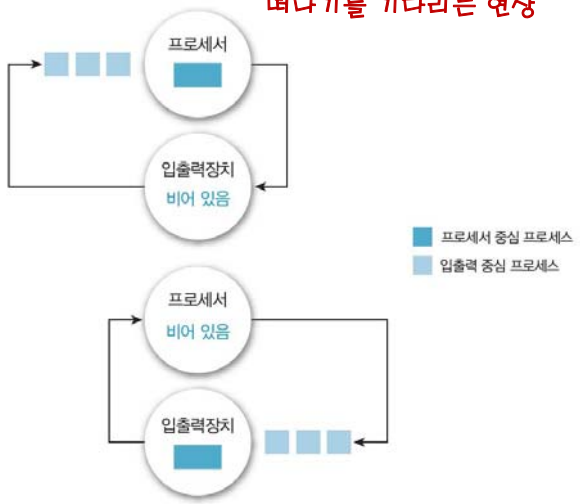


그림 6-18 호위 효과

1. 선입선처리 스케줄링

- 선입선처리 스케줄링의 장점과 단점

표 6-3 선입선처리 스케줄링의 장점과 단점

장점	• 스케줄링의 이해와 구현이 단순하다. • 준비 큐에 있는 모든 프로세스가 결국 실행되므로 기아 없는 공정한 정책이다. • 프로세서가 지속적으로 유용한 프로세스를 수행하여 처리율이 높다.
단점	• 비선점식이므로 대화식 프로세스(작업)에는 부적합하다. • 장기 실행 프로세스가 뒤의 프로세스(작업)를 모두 지연시켜 평균 대기시간이 길어져 최악의 대기시간이 된다. • 긴 프로세스(작업)가 실행되는 동안 짧은 프로세스(작업)가 긴 대기시간으로 호위 효과가 발생할 수 있다.

2. 최소작업 우선 스케줄링 SJF, Shortest Job First

- 최소작업 우선 스케줄링의 개념
  - 각 작업의 프로세서 실행 시간 이용하여 **프로세서가 사용 가능할 때 실행 시간이 가장 짧은 작업(프로세스)에 할당하는 방법**
  - 원리

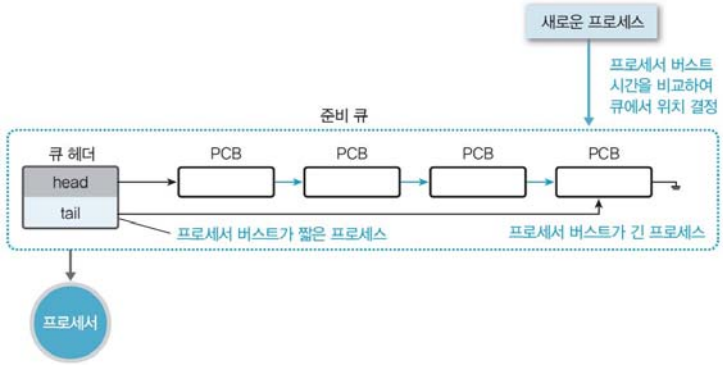


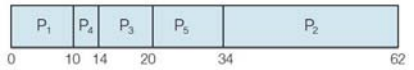
그림 6-19 최소작업 우선 스케줄링의 원리

2. 최소작업 우선 스케줄링 SJF, Shortest Job First

- 최소작업 우선 스케줄링 예

프로세스	도착 시간	실행 시간
P_1	0	10
P_2	1	28
P_3	2	6
P_4	3	4
P_5	4	14

(a) 준비 큐



(b) 간트 차트

그림 6-21 최소작업 우선 스케줄링 예

표 6-4 그림 6-21 예의 반환시간과 대기시간

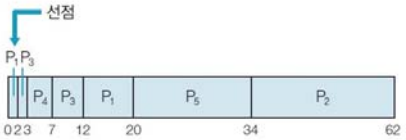
프로세스	반환시간	대기시간
P_1	10	0
P_2	(62 − 1) = 61	(34 − 1) = 33
P_3	(20 − 2) = 18	(14 − 2) = 12
P_4	(14 − 3) = 11	(10 − 3) = 7
P_5	(34 − 4) = 30	(20 − 4) = 16
평균 반환시간 : 26[= (10 + 61 + 18 + 11 + 30)/5]		평균 대기시간 : 13.6[= (0 + 33 + 12 + 7 + 16)/5]

2. 최소작업 우선 스케줄링 SRTF, Shortest Remaining Time First

▪ 선점 최소작업 우선 스케줄링 예

프로세스	도착 시간	실행 시간
P_1	0	10
P_2	1	28
P_3	2	6
P_4	3	4
P_5	4	14

(a) 준비 큐



(b) 간트 차트

그림 6-22 선점 최소작업 우선 스케줄링 예

표 6-5 그림 6-22 예의 반환시간과 대기시간

프로세스	반환시간	대기시간
P_1	(20-0) = 20	(12 – 2) = 10
P_2	(62 – 1) = 61	(34 – 1) = 33
P_3	(12-2) = 10	(7 – 3) = 4
P_4	(7 – 3) = 4	(3 – 3) = 0
P_5	(34 – 4) = 30	(20 – 4) = 16
평균 반환시간 : 25 [=(20+61+10+4+30)/5]		평균 대기시간 : 12.6[=(10 + 33 + 4 + 0 + 16)/5]

2. 최소작업 우선 스케줄링 SJF, Shortest Job First

▪ 최소작업 우선 스케줄링 : 짧은 작업 우선 처리 예

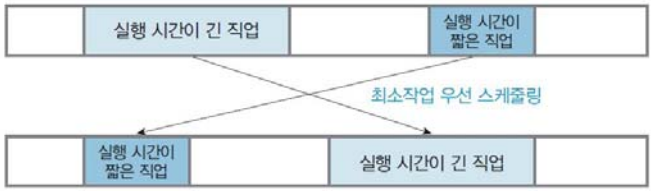


그림 6-23 최소작업 우선 스케줄링 : 짧은 작업 우선 처리

▪ 최소작업 우선 스케줄링의 장점과 단점

표 6-6 최소작업 우선 스케줄링의 장점과 단점

장점	항상 실행 시간이 짧은 작업을 신속하게 실행하므로 평균 대기시간이 가장 짧다.
단점	• 초기의 긴 작업을 짧은 작업을 종료할 때까지 대기시켜 기아가 발생한다. • 기본적으로 짧은 작업이 항상 실행되도록 설정하므로 불공정한 작업을 실행한다. • 실행 시간을 예측하기가 어려워 실용적이지 못하다.

3. 우선 순위 스케줄링 priority scheduling

▪ 우선순위 스케줄링의 개념

- 우선순위가 동일한 프로세스들은 선입선처리 순서로 스케줄링
- 원리

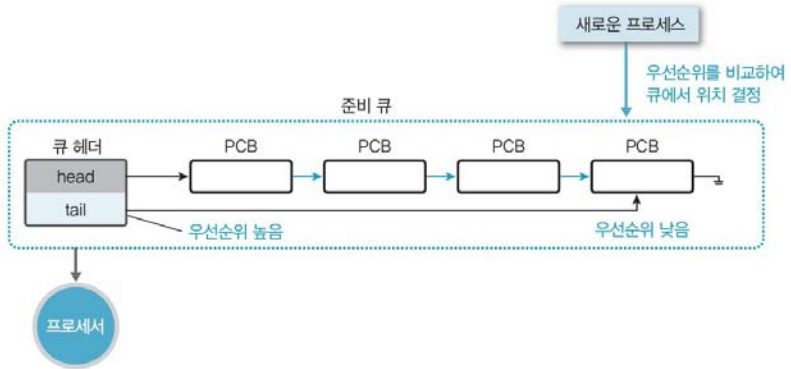


그림 6-24 우선순위 스케줄링

3. 우선 순위 스케줄링 priority scheduling

- 실행 시간이 클수록 우선순위가 낮고, 우선순위는 0~7 또는 0~4,095 범위의 수 사용
- 우선순위를 내부적 또는 외부적으로 정의
  - 내부적 우선순위 : 제한 시간, 기억장소 요청량, 사용 파일 수, 평균 프로세서 버스트에 대한 평균 입출력 버스트의 비율 등
  - 외부적 우선순위 : 프로세스 중요성, 사용료 많이 낸 사용자, 작업을 지원하는 부서, 정책적인 요인 등
- 우선순위 스케줄링도 선점 또는 비선점 가능
- 4단계 우선순위를 갖는 스케줄링 알고리즘

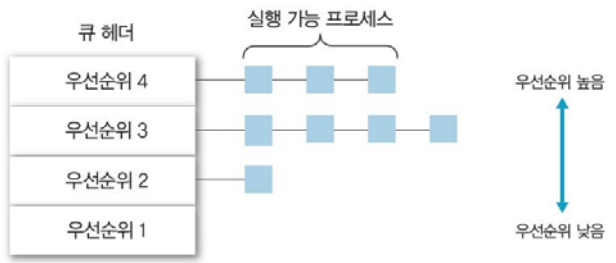


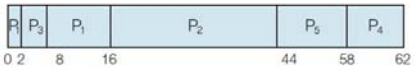
그림 6-25 네 단계 우선순위를 갖는 스케줄링 알고리즘

3. 우선 순위 스케줄링priority scheduling

- **우선 순위 스케줄링 예** ※우선순위 값이 클수록 우선순위가 높다고 가정

프로세스	도착 시간	실행 시간	우선순위
P_1	0	10	3
P_2	1	28	2
P_3	2	6	4
P_4	3	4	1
P_5	4	14	2

(a) 준비 큐



(b) 선점 우선순위 간트 차트

(c) 비선점 우선순위 간트 차트

그림 6-26 우선순위 스케줄링 예

3. 우선 순위 스케줄링priority scheduling

- **우선 순위 스케줄링 예**

표 6-7 그림 6-26에서 (b) 선점 예의 반환시간과 대기시간

프로세스	반환시간	대기시간
P_1	(16-0) = 16	(8 – 2) = 6
P_2	(44 – 1) = 43	(16 – 1) = 15
P_3	(8 – 2) = 6	(2 – 2) = 0
P_4	(62-3) = 59	(58 – 3) = 55
P_5	(58-4) = 54	(44 – 4) = 40
평균 반환시간 : 35.6 [=(16+43+6+59+64)/5]		평균 대기시간 23.2 (6 + 15 + 0 + 55 + 40)/5]

표 6-8 그림 6-26에서 (c) 비선점 예의 반환시간과 대기시간

프로세스	반환시간	대기시간
P_1	10	0
P_2	(44 – 1) = 43	(16 – 1) = 15
P_3	(16 – 2) = 14	(10 – 2) = 8
P_4	(62 – 3) = 59	(58 – 3) = 55
P_5	(58 – 4) = 54	(44 – 4) = 40
평균 반환시간 : 36[=(10 + 43 + 14 + 59 + 54)/5]		평균 대기시간 : 23.6[=(0 + 15 + 8 + 55 + 40)/5]

3. 우선 순위 스케줄링priority scheduling

- **우선 순위 스케줄링의 장점과 단점**

표 6-9 우선순위 스케줄링의 장점과 단점

장점	• 각 프로세스의 상대적 중요성을 정확히 정의할 수 있어 좋다. • 다양한 반응으로 실시간 시스템에 사용 가능하다.
단점	높은 우선순위 프로세스가 프로세서를 많이 사용하면 우선순위가 낮은 프로세스는 무한정 연기되는 기아가 발생할 수 있다.

3. 우선 순위 스케줄링priority scheduling

- **우선 순위 스케줄링의 문제**
  - 무한 정지와 기아
  - 해결 방법 : 에이징
    - 에이징
      - 시스템에서 오래 대기하는 프로세스들의 우선순위를 점진적으로 증가시키는 방법
      - 시간이 지나면 점차 프로세스의 우선순위 높아짐

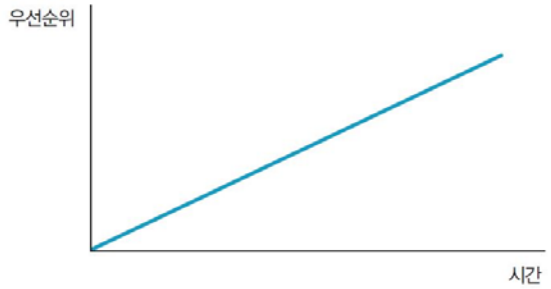
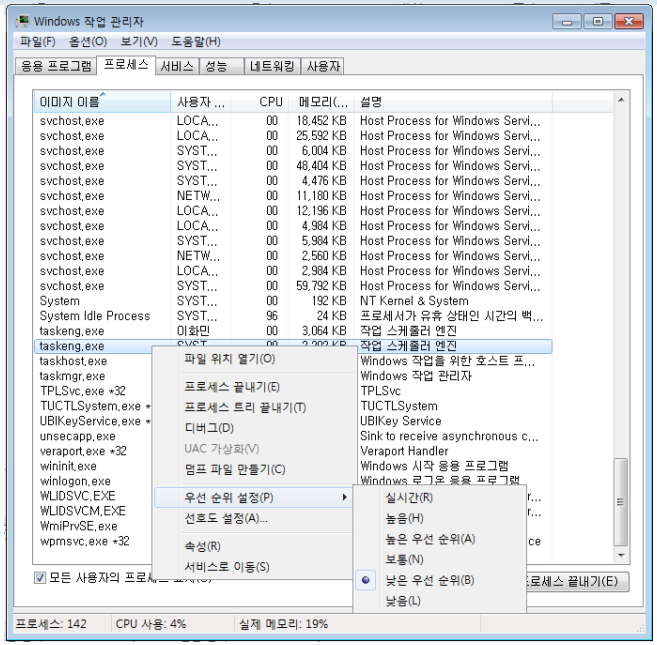


그림 6-27 에이징

Windows 우선순위



4. 라운드 로빈RR, round-robin 스케줄링

■ 라운드 로빈 스케줄링의 개념

- 특별히 시분할 시스템을 위해 설계
- 작은 단위의 시간인 규정 시간량time quantum 또는 시간 할당량time slice 정의
- 보통 규정 시간량은 10×10밀리초에서 100×10밀리초 범위
- 준비 큐를 순환 큐circular queue로 설계하여 스케줄러가 준비 큐를 돌아가면서 한 번에 한 프로세스에 정의된 규정 시간량만큼 프로세서 제공

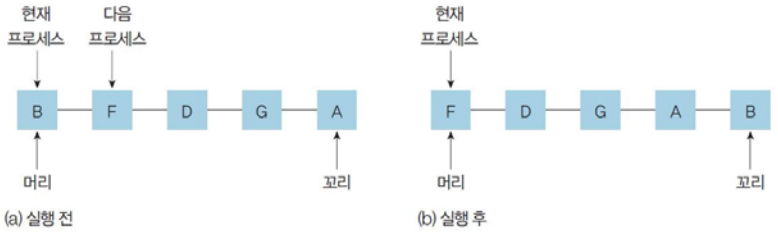


그림 6-28 프로세스 B를 실행한 후의 준비 큐

4. 라운드 로빈round-robin 스케줄링

- 원리

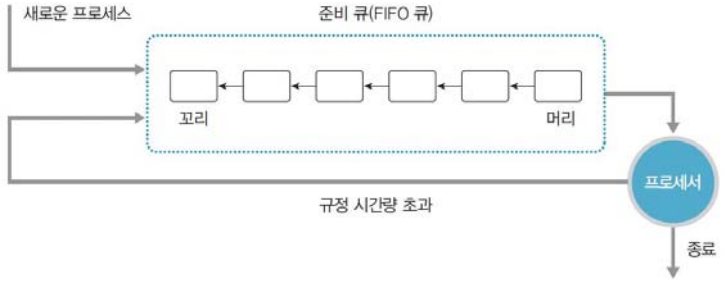


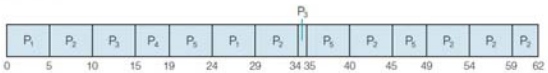
그림 6-29 라운드 로빈 스케줄링

4. 라운드 로빈round-robin 스케줄링

■ 라운드 로빈 스케줄링 예

프로세스	도착 시간	실행 시간
P_1	0	10
P_2	1	28
P_3	2	6
P_4	3	4
P_5	4	14

(a) 준비 큐



(b) 간트 차트

그림 6-30 라운드 로빈 스케줄링 예

표 6-10 그림 6-30 예의 반환시간과 대기시간

프로세스	반환시간	대기시간	
P_1	29	(24-5)=19	
P_2	(62-1)=61	(49-15-1)=33	(5-1)+(29-10)+(40-34)+(49-45) =33
P_3	(35-2)=33	(34-5-2)=27	(10-2)+(34-15) =27
P_4	(19-3)=16	(15-3) = 12	
P_5	(49-4)=45	(45-10-4)=31	(19-4)+(35-24)+(45-40) = 31
평균 반환시간 : 36.8[=(29+61+33+16+45)/5]		평균 대기시간 :	24.4 [=(19+33+27+12+31)/5]

응답(반응)시간
0
(5-1) = 4
(10-2) = 8
(15-3) = 12
(19-4) = 15
7.8 [(0+4+8+12+15)/5]

4. 라운드 로빈round-robin 스케줄링

- 문맥 교환 시간이 라운드 로빈 스케줄링에 미치는 영향

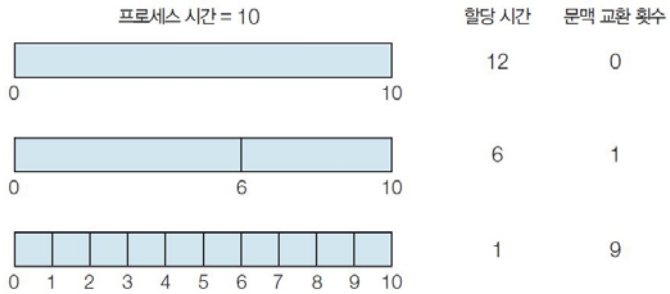


그림 6-31 문맥 교환 횟수를 증가시키는 작은 규정 시간량

※규정 시간량이 작을수록 문맥 교환 횟수는 많아지므로 문맥 교환에 소요하는 시간보다 규정 시간량을 충분히 크게 해야 함

4. 라운드 로빈round-robin 스케줄링

- 규정 시간량에 따른 반환 시간의 변화

프로세스	시간
P_1	6
P_2	3
P_3	1
P_4	7

※규정 시간량이 작으면 문맥 교환을 많이하므로 평균 반환시간이 더 좋지 않다

그림 6-32 규정 시간량에 따른 반환시간의 변화

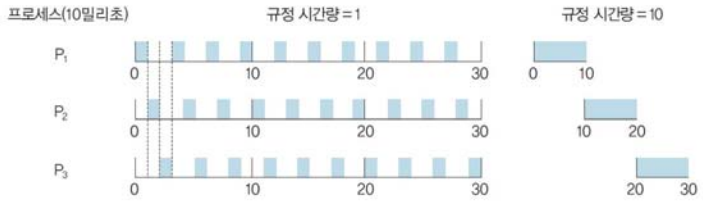


그림 6-33 규정 시간량에 따른 프로세스의 반환시간

5. 다단계 큐 스케줄링MLQ, MultiLevel Queue

- 다단계 큐 스케줄링의 개념
  - 각 작업을 서로 다른 묶음으로 분류할 수 있을 때 사용
  - 준비 상태 큐를 종류별로 여러 단계로 분할, 그리고 작업을 메모리의 크기나 프로세스의 형태에 따라 특정 큐에 지정. 각 큐는 자신만의 독자적인 스케줄링 갖음
  - 원리

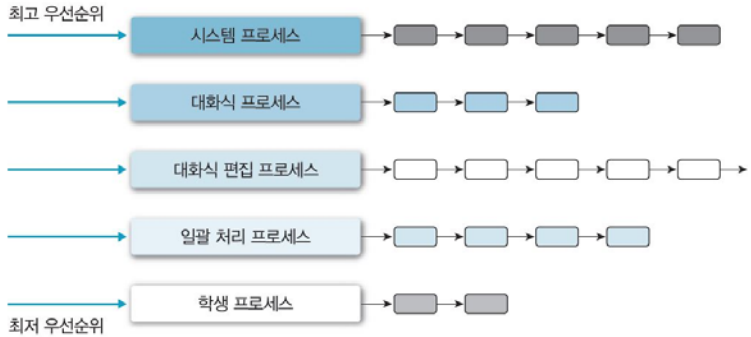


그림 6-34 다단계 큐 스케줄링

각 큐는 순서대로 절대적인 우선순위 갖음
큐 사이에 시간을 나눠 사용할 수도 있음

5. 다단계 큐 스케줄링MLQ, MultiLevel Queue

- 다단계 큐 스케줄링의 장점과 단점

표 6-12 다단계 큐 스케줄링의 장점과 단점

장점	응답이 빠르다.
단점	• 여러 준비 큐와 스케줄링 알고리즘 때문에 추가 오버 헤드가 발생한다. • 우선순위가 낮은 큐의 프로세스는 무한정 대기하는 기아가 발생할 수 있다.

6. 다단계 피드백 큐MLFQ, MultiLevel Feedback Queue스케줄링

■ 다단계 피드백 큐 스케줄링의 개념

- 작업이 시스템에 들어가면 한 큐에서만 고정 실행
- 스케줄링 부담이 적다는 장점이 있으나 융통성이 떨어진다는 단점
- 작업이 큐 사이 이동 가능(프로세서 버스트의 특성에 따라 분리)
- 구조

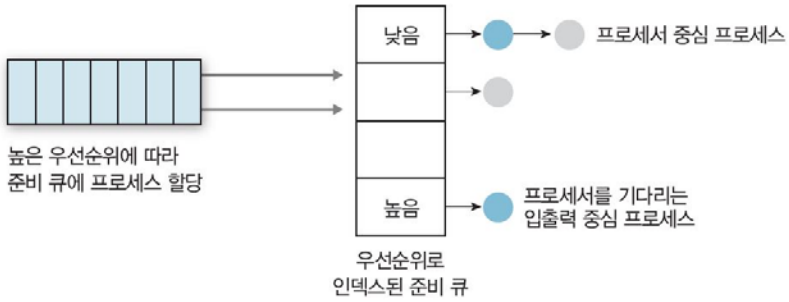


그림 6-35 다단계 피드백 큐 스케줄링 구조

6. 다단계 피드백 큐MLFQ, MultiLevel Feedback Queue스케줄링

- 원리

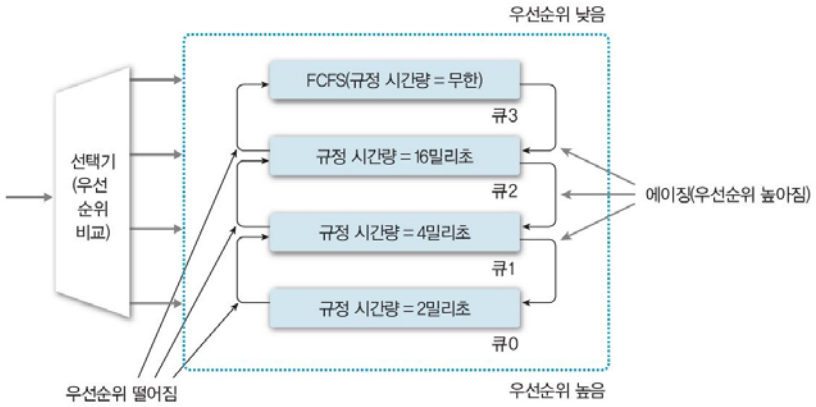


그림 6-36 다단계 피드백 큐 스케줄링

6. 다단계 피드백 큐MLFQ, MultiLevel Feedback Queue스케줄링

■ 다단계 피드백 큐 스케줄링의 정의 방법

- 큐queue 수
- 각 큐에 대한 스케줄링
- 작업을 좀 더 높은 우선순위의 큐로 격상시키는 시기를 결정하는 방법
- 작업을 좀 더 낮은 우선순위의 큐로 격하시키는 시기를 결정하는 방법
- 프로세스들이 어느 큐에 들어갈 것인지 결정하는 방법
- •프로세스가 서비스를 받는 시기를 결정하는 방법

6. 다단계 피드백 큐MLFQ, MultiLevel Feedback Queue스케줄링

■ 다단계 피드백 큐와 라운드 로빈 스케줄링의 비교

프로세스	실행 시간
P_1	30
P_2	20
P_3	10

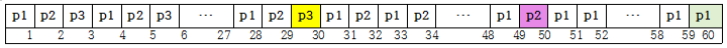
(a) 준비 큐

(b) 간트 차트

※ 라운드로빈 : TQ = 1
다단계 피드백 큐 TQ =1/2/4

그림 6-37 다단계 피드백 큐와 라운드 로빈(순환 할당) 스케줄링의 비교 예
표 6-13 그림 6-37 예의 반환시간과 대기시간

(a) 반환시간



프로세스	라운드 로빈RR 스케줄링	다단계 피드백 큐MLFQ 스케줄링
P_1	60	60
P_2	50	53
P_3	30	52
평균 반환시간 : $46\frac{2}{3}$[=(60+50+30)/3]		평균 반환시간 : $48\frac{1}{3}$[=(60+53+32)/3]

(b) 대기시간

프로세스	라운드 로빈RR 스케줄링	다단계 피드백 큐MLFQ 스케줄링
P_1	30	(53−23)=30
P_2	30	(52−19)=33
P_3	20	(29−7)=22
평균 대기시간 : 30[=(30+30+30)/3]		평균 대기시간 : $28\frac{1}{3}$[=(30+33+22)/3]

6. 다단계 피드백 큐MLFQ, MultiLevel Feedback Queue스케줄링

▪ 다단계 피드백 큐 스케줄링의 장점과 단점

표 6-14 다단계 피드백 큐 스케줄링의 장점과 단점

장점	• 매우 유연하여 스케줄러를 특정 시스템에 맞게 구성할 수 있다. • 자동으로 입출력 중심과 프로세서 중심 프로세스를 분류한다. • 적응성이 좋아 프로세스의 사전 정보가 없어도 최소작업 우선 스케줄링의 효과를 보여 준다.
단점	설계와 구현이 매우 복잡하다.

7. HRNHighest Response-ratio Next 스케줄링

▪ HNR 스케줄링의 개념

- 최소작업 우선 스케줄링의 약점인 긴 작업과 짧은 작업 간의 지나친 불평 등을 보완
- 비선점 스케줄링이며 우선순위 스케줄링의 또 다른 예
- **선입선처리 스케줄링과 최소작업 우선 스케줄링의 약점을 해결하기 위해 제안**
- **우선순위**

$$\text{우선순위} = \frac{\text{서비스를 받을 시간} + \text{대기한 시간}}{\text{서비스를 받을 시간}}$$

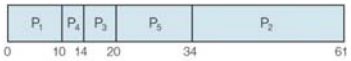
7. HRNHighest Response-ratio Next 스케줄링

▪ HNR 스케줄링의 예

프로세스	도착 시간	실행 시간
P_1	0	10
P_2	1	28
P_3	2	6
P_4	3	4
P_5	4	14

(a) 준비 큐

$$\text{우선순위} = \frac{\text{서비스를 받을 시간} + \text{대기한 시간}}{\text{서비스를 받을 시간}}$$



(b) 간트 차트

그림 6-40 HRN 스케줄링 예

표 6-15 그림 6-40 예의 반환시간과 대기시간

프로세스	반환시간	대기시간
P_1	10	0
P_2	(62 − 1) = 61	(34 − 1) = 33
P_3	(20 − 2) = 18	(14 − 2) = 12
P_4	(14 − 3) = 11	(10 − 3) = 7
P_5	(34 − 4) = 30	(20 − 4) = 16
평균 반환시간 : 26[=(10 + 61 + 18 + 11 + 30)/5]		평균 대기시간 : 13.6 [=(0+33+12+7+16)/5]

7. HRNHighest Response-ratio Next 스케줄링

▪ HNR 스케줄링의 장점과 단점

표 6-16 HRN 스케줄링의 장점과 단점

장점	• 자원을 효율적으로 활용한다. • 기아가 발생하지 않는다.
단점	오버헤드가 높을 수 있다(메모리와 프로세서 낭비).

8. 다중 프로세서 스케줄링

■ 강결합된 공유 메모리 구조에서 스케줄링 방법

❶ 프로세서 자신이 스스로 스케줄링

- 각 프로세서는 공통의 준비 상태 큐에서 실행할 프로세 스 하나를 선택
- 프로세서 2개가 같은 프로세스를 선택하지 않도록 해야 하고, 프로세스가 큐에서 누락되지 않도록 해야 함
- 단일 프로세서 스케줄링(우선순위, 순환 할당 등) 방법 활용할 수 있으나 오히려 스케줄링이 복잡하여 오버헤드 증가 가능

❷ 비대칭 다중 처리AMP, Asymmetric MultiProcessing는 한 프로세서를 다른 모든 프로세서의 스케줄 러로 지정, 주종Master/Slave 구조를 보임

- 주 프로세서는 프로세스들을 스케줄링하여 프로세스를 활성화
- 종 프로세스에 입출력 호출 등의 서비스가 필요하면 주 프로세서에 요청하여 서비스의 처리 기다림
- 이런 방법은 앞서 다룬 단일 프로세서 다중 프로그래밍 방법과 비슷함
- 주 프로세서의 오류는 시스템 전체 정지시키며, 주 프로세서의 과중한 오버헤드는 성능의 병목 지점이 될 수 있는 문제점

9. 스레드 스케줄링

■ 스레드 스케줄링의 개념

- 스레드를 이용하여 응용 프로그램을 동일한 주소 공간에서 동시에 실행하고 협동하는 스레드들로 구현할 수 있다는 것. 즉, 입출력과 프로세서 처리 중첩 가능
- 스레드 문맥 교환은 프로세스 문맥 교환보다 오버헤드 적어 응용 프로그램 하나의 여러 스레드를 동시에 다른 프로세서에서 실행한다면 성능 현저하게 향상 가능.
- 스레드 간에 많은 상호작용을 요청하는 응용 프로그램에서는 성능에 큰 영향을 줄 수 있음

■ 다중 프로세서 스레드 스케줄링과 프로세서 할당에 대한 일반적인 방법

부하 공유load sharing

- 프로세서를 특정 프로세스 하나에 할당하지 않고 전역 큐에서 프로 세서 유지. 그리고 쉬고 있는 프로세스는 전역 큐에서 스레드 한 개를 선택. 단일 프로세서 환경에서 사용한 방법을 그대로 채택한 가장 단순한 방법

9. 스레드 스케줄링

갱gang 스케줄링

- 관련된 스레드의 집합을 일대일 대응 원칙에 따라 프로세서 집합에서 동시에 실행할 수 있도록 스케줄링하는 방법
- 밀접하게 관련된 프로세스들이 병렬로 실행된다면 동기화 대기 및 프로세스 문맥 교환의 횟수 최소화하여 성능 향상

전용 프로세서 할당

- 부하 공유와는 반대로 스레드들을 실행 전담 프로세서에 할당하여 정의된 스케줄링 제공하는 방법
- 각 프로그램은 실행되는 동안 프로그램의 스레드수와 동일한 수의 프로세스 할당받기 때문에 프로세스가 낭비

동적 스케줄링

- 프로그램의 스레드 수는 실행 도중 변화 가능, 프로세스의 스레드 수를 동적으로 변경하여 운영체제가 시스템 이용률을 높일 수 있도록 부하 조절을 허용한 방법

Section 03 스케줄링 알고리즘의 평가

■ 평가 방법 : 분석적(해석적) 평가

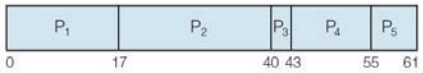
- 분석적 평가
  - 작업 부하를 줄이는 데 알고리즘의 성능을 평가하는 공식, 값을 생성하는 알고리즘, 시스템 작업 부하 이용
  - 사전에 정의된 특 정한 작업에서 각 알고리즘의 성능 평가
  - 예

프로세스	버스트 시간
P_1	17
P_2	23
P_3	3
P_4	12
P_5	6

그림 6-41 스케줄링 알고리즘 평가 예

2. 스케줄링 알고리즘의 평가 예

- 선입 선처리 스케줄링의 평가

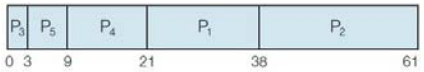


(a) 간트 차트

프로세스	대기시간
P_1	0
P_2	17
P_3	40
P_4	43
P_5	55
평균 대기시간 : 31[=(0+17+40+43+55)/5]	

(b) 대기시간

- 비선점 최소작업 우선 스케줄링의 평가



(a) 간트 차트

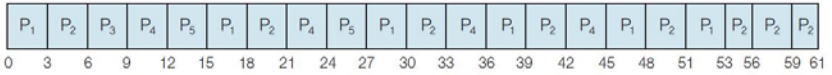
프로세스	대기시간
P_1	21
P_2	38
P_3	0
P_4	9
P_5	3
평균 대기시간 : 14.2[=(21+38+0+9+3)/5]	

(b) 대기시간

그림 6-43 비선점 최소작업 우선 스케줄링 평가

2. 스케줄링 알고리즘의 평가 예

- 라운드 로빈 스케줄링의 평가



(a) 간트 차트

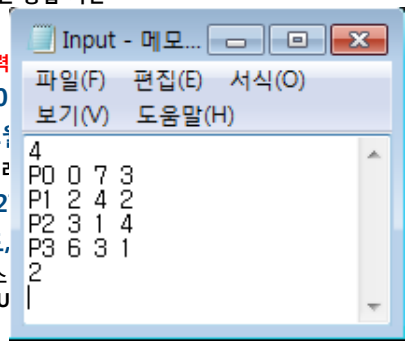
프로세스	대기시간
P_1	(51−5−5−5)=36
P_2	(53−5−5−5)=38
P_3	6
P_4	(42−3−3−3)=33
P_5	(24−3)=21
평균 대기시간 : 26.8[=(36+38+6+33+21)/5]	

(b) 대기시간

그림 6-44 라운드 로빈 스케줄링 평가

OS Term Project

- 7가지 CPU 스케줄링 시뮬레이터 개발
- FCFS, SJF, SRT, 비선점 Priority, 선점 Priority, RR, HRN
- 입력(반드시 순서 지킬 것/우선순위 숫자가 작을수록 우순순위 높게)
  - (프로세스 수), 프로세스ID, 도착시간, 서비스시간, 우선순위, 시간할당량
- 출력(반드시 순서 지킬 것)
  - 간트 차트, 각 프로세스별 대기시간 및 평균 대기 시간, 각 프로세스별 반환시간 및 평균 반환 시간, 각 프로세스별 응답시간 및 평균 응답 시간
- 개발 언어 : C, C++, Java, C#
  - 콘솔 창 실행으로 개발할 경우 반드시 입력
- 팀원 : 반드시 3명으로 구성할 것(10
- Term Project 계획서 제출기한 : 11
  - 팀 구성, 팀원 역할, 개발언어, UI, 함수/클
- Term Project 발표 및 데모 : 11월 2
- Term Project 최종보고서, 소스코드,
  - 팀 구성, 팀원 역할, 개발언어, 함수/클래스
    각 알고리즘 핵심부분 코드 설명, 개발한 U
    느낀 점 등



Thank You