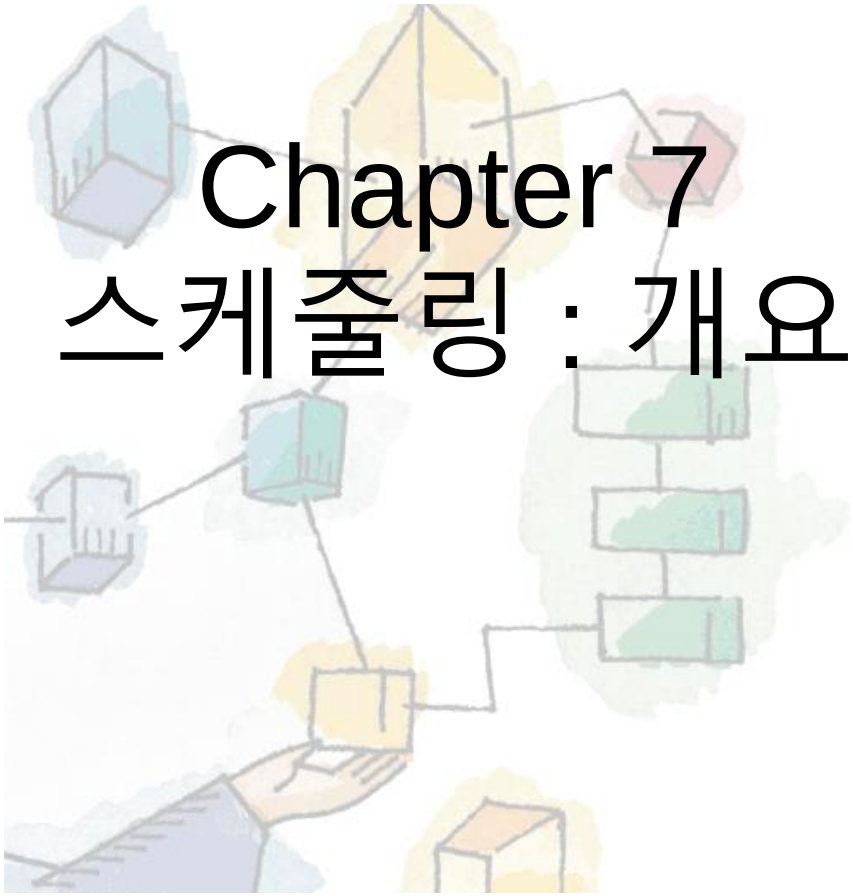
Chapter-2

운영 체제

정내훈

2023년 가을학기
게임공학과
한국공학대학교

Chapter 7
스케줄링 : 개요



스케쥴링 : 개요

- 여러 개의 프로세스가 있는데 어느 프로세스를 지금 실행해야 하는가? (Scheduler가 누구를 선택하는가?)
- 목표
  - 어차피 총 실행 시간은 정해져 있지 않은가?
    - 누구를 먼저 끝낼 것인가도 중요하다.
  - 공정해야 한다.
  - 가능하면 많은 프로세스가 일찍 종료해야 한다.
    - **반환시간**의 합이 작아야 한다.
- 작업 (Work Load)
  - 멈춤없이 실행되는 실행구간
  - CPU만 사용한다. (IO요청을 작업 종료로 본다. 실행/준비 상태의 프로세스만 스케줄링)
  - 구간의 크기를 미리 알 수도 있고, 모를 수도 있다.
  - 스케줄링은 작업단위로 이루어진다.

반환시간

스케쥴링 평가항목

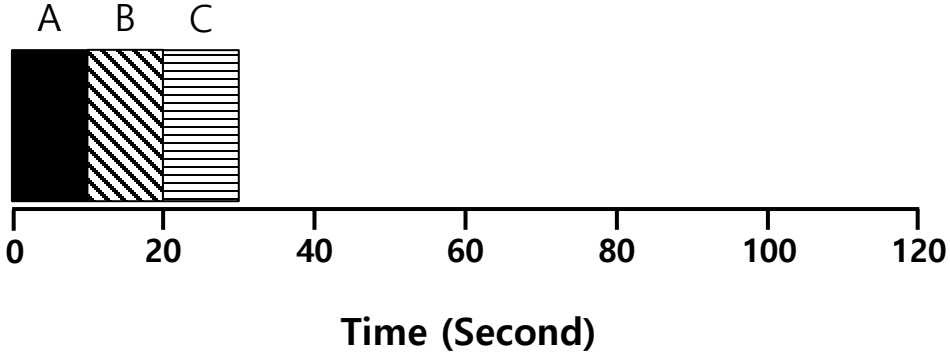
- 성능 평가 기준 : 반환 시간 (Turnaround time)
  - 작업이 완료된 시간에서 작업이 도착한 시간을 뺀 시간
  - 전체 반환시간이 작을수록 좋다. (평균 반환시간)

$$T_{\text{반환시간}} = T_{\text{완료시간}} - T_{\text{도착시간}}$$

- 다른 평가 기준 : 공정성(fairness).
  - 성능과 대립할 수 있다.

선입 선출(FIFO)

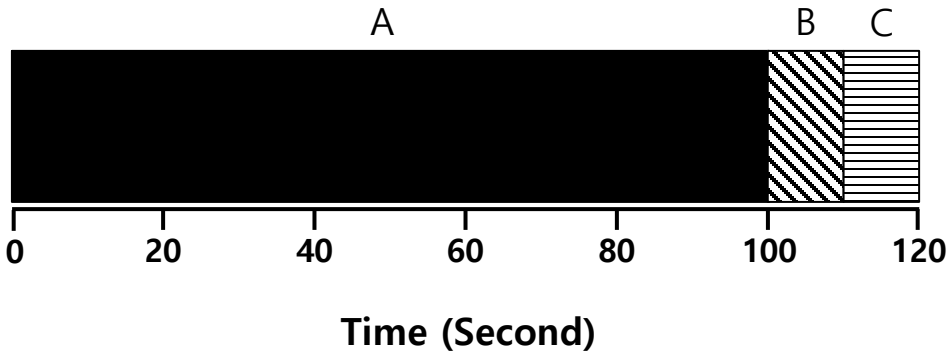
- First Come, First Served (FCFS)
  - 매우 간단하고 구현하기 쉽다.
- Example:
  - A, B, C가 순서대로 동시에 도착했을 때
  - 각 작업은 10초 씩 실행.



$$\text{평균 반환 시간} = \frac{10 + 20 + 30}{3} = 20 \text{ 초}$$

선입선출의 단점 : 호위효과(Convoy effect)

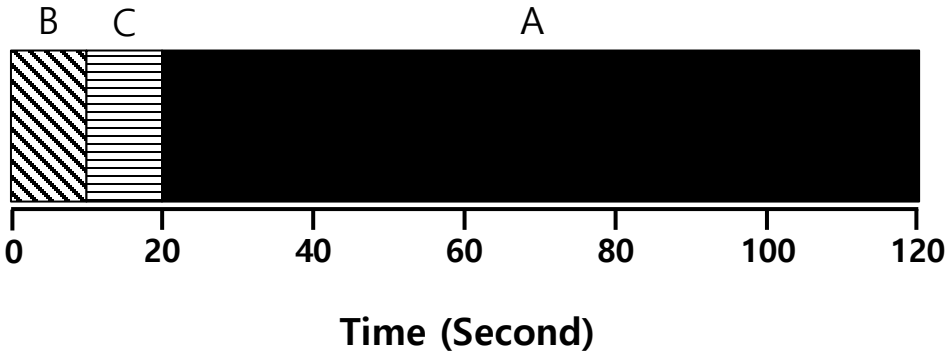
- 작업 마다 실행 시간이 다를 경우
- 예:
  - A, B, C가 순서대로 동시에 도착.
  - A는 100초, B와 C는 10초간 실행.



$$\text{평균 반환 시간} = \frac{100 + 110 + 120}{3} = 110 \text{ 초}$$

최단 작업 우선 (SJF)

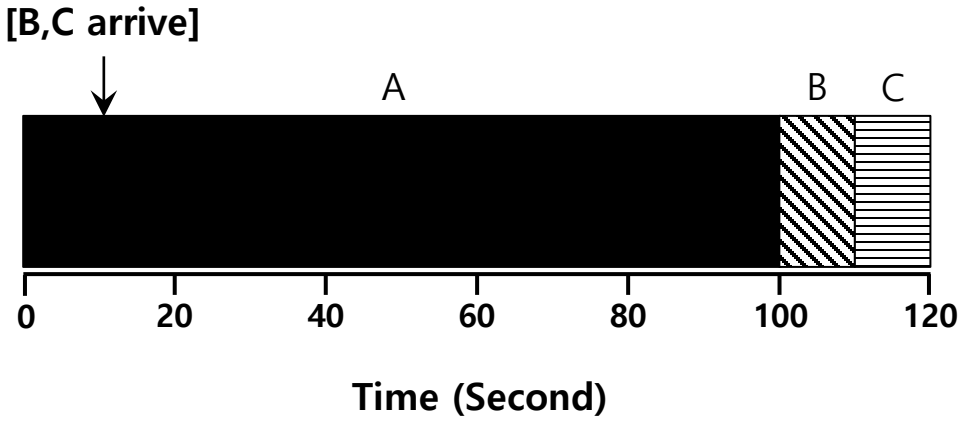
- 가작 짧은 작업 먼저, 다음에는 두번째 짧은 작업, ..
  - 비선점(Non-preemptive) 스케줄러중 최선
- 예:
  - A, B, C가 순서대로 동시 도착.
  - A가 100초, B와 C가 10초 실행.



$$\text{평균 반환 시간} = \frac{10 + 20 + 120}{3} = 50 \text{ 초}$$

B, C가 늦게 도착한 경우

- Example:
  - A 가 t=0에 도착해서 100초간 실행.
  - B와 C가 t=10에 도착해서 10초간 실행해야 한다면.



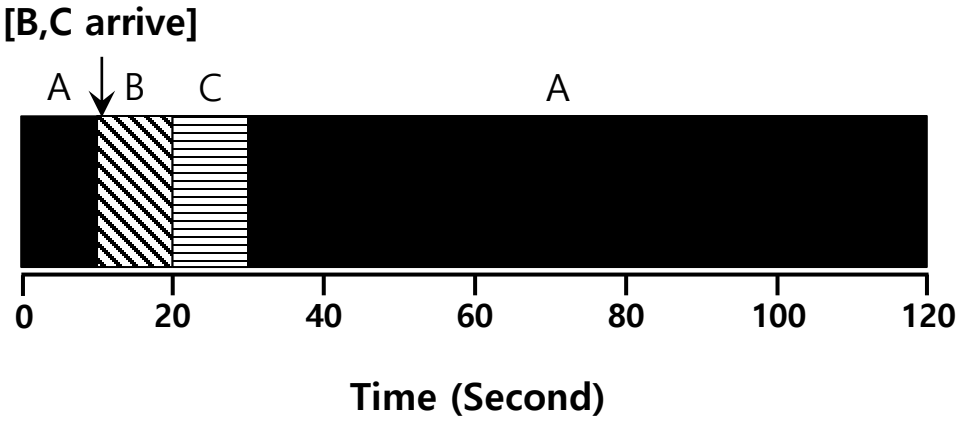
$$\text{평균 반환 시간} = \frac{\mathbf{100} + (\mathbf{110} - \mathbf{10}) + (\mathbf{120} - \mathbf{10})}{\mathbf{3}} = \mathbf{103.33}\ \text{초}$$

최소 잔여시간 우선 (STCF)

- 잔여 시간이 제일 작은 작업을 우선 실행
- Shortest Remaining Time First(SRF)
- SJF에 선점(preemption) 추가
  - 강제 작업 전환 필요
  - 다른 이름 Preemptive Shortest Job First (PSJF)
- 새로운 작업이 시작되면:
  - 새 작업과 기존의 작업들의 잔여시간 조사.
  - 잔여시간이 가장 작은 작업을 스케쥴

최소 잔여시간 우선(STCF)

- Example:
  - A arrives at t=0 and needs to run for 100 seconds.
  - B and C arrive at t=10 and each need to run for 10 seconds



$$\text{평균 반환 시간} = \frac{(120-0)+(20-10)+(30-10)}{3} = 50 \text{ 초}$$

응답시간

새로운 평가 기준: 응답 시간

- 작업이 도착한 시간부터 처음 스케줄 될 때까지의 시간.

$$T_{응답시간} = T_{최초실행} - T_{도착시간}$$

  - STCF관련 기법들은 응답시간이 좋지 않음

어떻게 하면 응답시간을 위주로 스케줄링을 할 수 있는가?

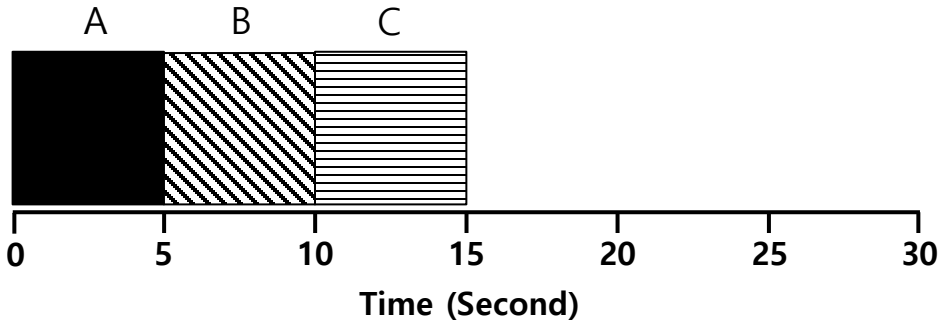
라운드 로빈 (RR)

- 타임 슬라이스 스케쥴링
  - 작업을 실행하고 타임 슬라이스 시간이 지나면 **준비 큐**에 있는 다음 작업으로 문맥 교환.
    - 타임 슬라이스를 타임 퀀텀, 스케쥴링 퀀텀이라고 도 함.
  - 위의 과정을 작업이 종료될 때 까지 반복
  - 타임 슬라이스의 크기는 타이머-인터럽트 간격의 정수 배 이다.

RR은 공평하지만 반환 시간 같은 요소는 좋지 않다.

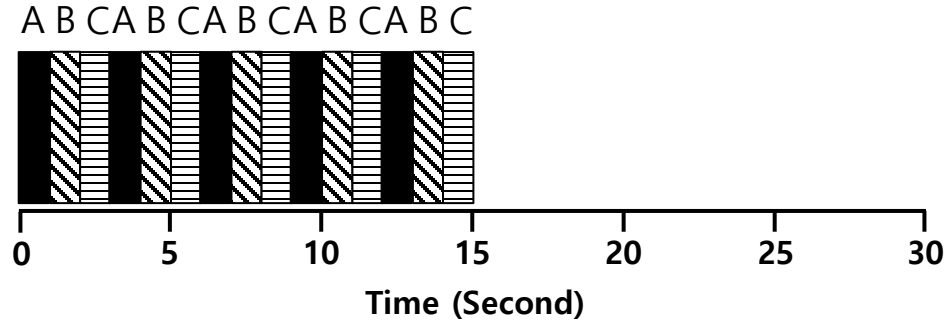
RR 스케쥴링 예제

- A, B와 C가 동시 도착.
- 모두 5초간 실행.



SJF (반응시간이 안좋음)

$$T_{\text{평균 반응}} = \frac{0+5+10}{3} = 5\text{초}$$



1초 타임슬라이스의 RR (좋은 반응시간)

$$T_{\text{평균반응}} = \frac{0+1+2}{3} = 1\text{초}$$

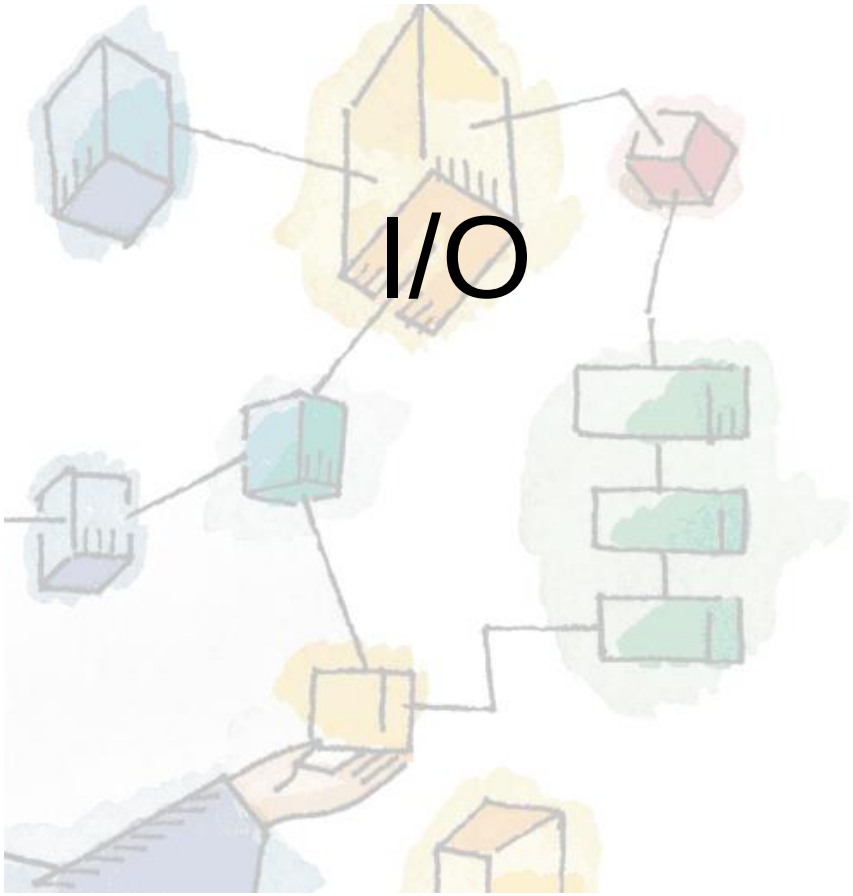
타임 슬라이스 크기가 중요

- 타임슬라이스가 작을 수록
  - 반응시간이 짧아진다.
  - 잦은 문맥교환으로 문맥 교환 비용이 전체 성능을 좌우한다.

- 타임슬라이스가 길수록
  - 교환 비용이 차지하는 비중이 줄어든다.
  - 반응시간이 길어진다.

시스템 설계자의 타임 슬라이스 크기 결정은 트레이드 오프 (trade-off)의 요소가 있다.

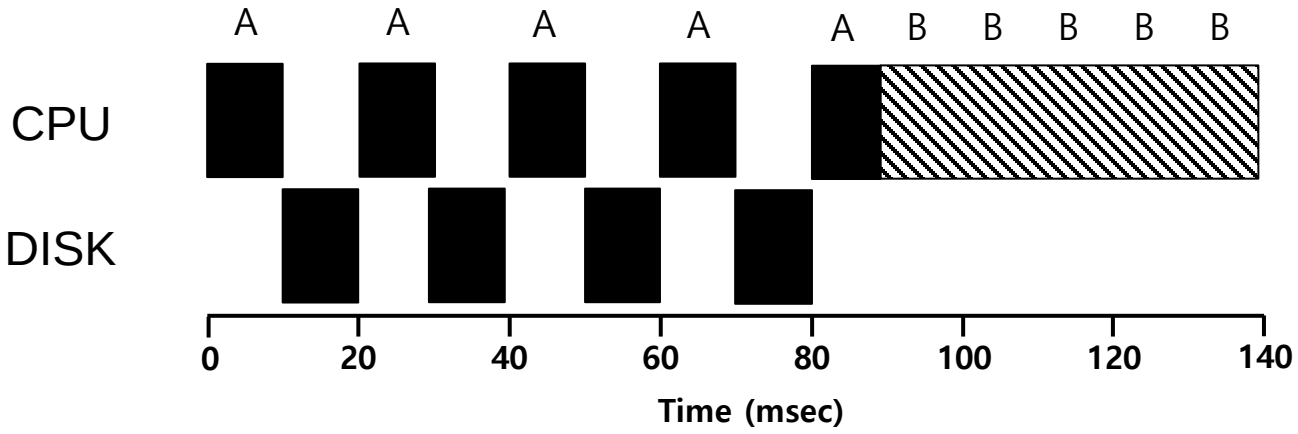
I/O



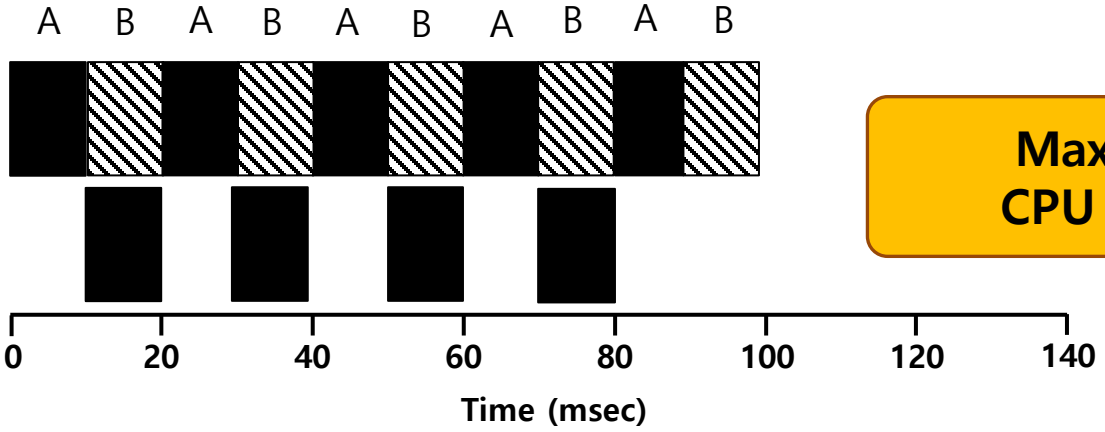
I/O 연산 고려

- 만약 I/O가 CPU를 사용하지 않는 다면?
  - DMA와 인터럽트의 조합으로 구현
  - CPU를 다른 프로세스가 사용하는 것이 바람직
- 보기:
  - A와 B는 CPU를 둘 다 50ms 사용
  - A는 10ms 실행할 때 마다 I/O 요청
    - 모든 I/O는 10ms 동안 실행
  - B는 I/O없이 50ms 계속 실행
  - 스케줄러가 A를 먼저 실행하고, A종료 후 B를 실행한다면?

I/O 연산 고려 (Cont.)



Poor Use of Resources



Maximize the CPU utilization

중첩(Overlapping)을 통한 더 나은 자원 활용

I/O 연산 고려 (Cont.)

- B를 먼저 실행했다면?
  - RR이 아닌 경우 작업 중첩이 일어나지 않는다.
- 작업이 I/O를 요청하면
  - I/O가 종료될 때 까지 작업은 대기상태가 된다.
  - 스케줄러는 다른 작업 에게 CPU를 할당해야 한다. **=> 중첩** (A와 B가 동시 실행)
- I/O가 종료 되면
  - 인터럽트로 운영체제가 실행 되면서 스케줄러가 동작한다.

요약

- 반환시간
  - 주어진 기간 동안 최대한 많은 작업을 종료 시키기
- 응답시간
  - 대화형 작업을 위한 평가 요소
- I/O실행
  - 중첩으로 전체 실행시간 감소
  - I/O중첩 개수? I/O 장치의 개수 * 작업 큐 길이
- 의문점?
  - 작업의 실행 시간을 미리 알 수 있는가?