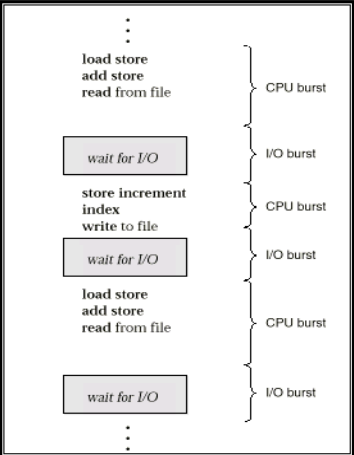
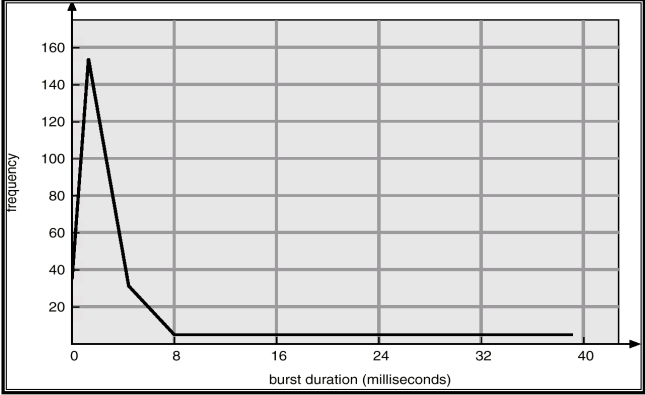
Ch5) Process Scheduling

1. CPU burst VS I/O burst



* Alternating Sequence of CPU and I/O bursts
* CPU burst: CPU가 명령어가 수행되는 구간 (Process가 running상태)
* I/O burst: wait상태에서 I/O가 끝날 때 까지 기다리는 구간(Process가 wait상태)
* CPU-bound process: CPU burst가 큰 프로세스(연산 위주)
* I/O-bound process: I/O burst가 큰 프로세스(I/O작업 위주)
* 우리가 쓰는 I/O-bound process에는 어떤 것들이 있을까? – 문서편집기(아래한글), 동영상 재생 프로그램 등등 대부분의 프로그램
* CPU-bound process에는 어떤 것들이 있을까? – 수퍼 컴퓨터의 기상예측 프로그램(계산이 많이 필요함) 등등 과학용 계산 프로그램. 그나마 우리가 쓰는 것에 가까운 것은 압축 프로그램(알집)



* CPU-burst time이 8ms 이하인 것을 알 수 있다.

1. CPU Scheduler

* CPU scheduling이란 short-term scheduling을 말한다.
* Long-term scheduling VS short-term scheduling
* Long-term scheduling: 옛날에 작은 메모리 환경일 때 어떤 프로세스를 메모리에 올려놓을지 결정하는 것(현재는 없어짐)
* Short-term scheduling: 그 메모리에 올라가 있는 동안, 프로세스 중에 어떤 것을 먼저 CPU에 올려놓고 처리할 것인지 결정하는 것(현재의 CPU scheduling)
* Dispatching VS Scheduling
* Dispatch – ready queue에 있는 프로세스를 CPU에 올리는 행위
* Scheduling – 어떠한 것을 CPU에 올릴지 계획
* 현재는 넓은 의미로 Scheduling으로 통합해서 사용함

1. Preemptive(선점) VS Non-preemptive(비선점)

* Preemptive – 중요도가 높은 프로세스가 등장하면 기존에 수행되던 프로세스를 중단하고 그 프로세스를 먼저 수행
* Non-preemptive – 더 중요한 프로세스가 들어오더라도 하던 것을 계속 수행
* Preemptive 스케줄링이 훨씬 효율적인 방법임. 대부분의 운영체제가 이 방식을 지원

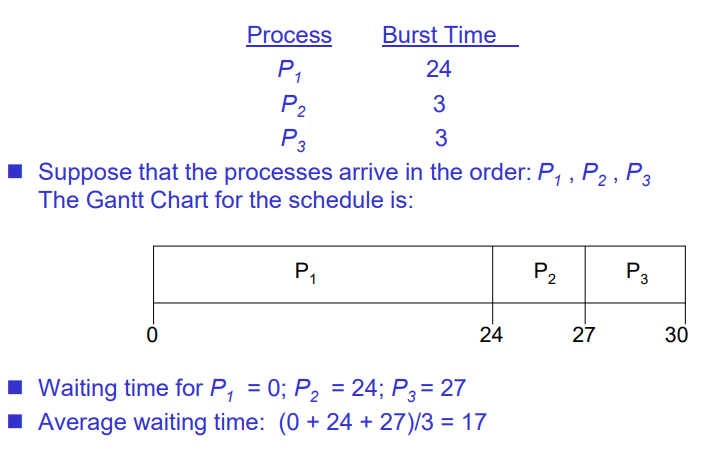
1. Scheduling Criteria
2. CPU utilization- CPU사용률
3. Throughput – 단위 시간당 처리량
4. Turnaround time – 총 수행 시간
5. Waiting time – Ready Queue에서 대기하는 시간(Device Queue에서 대기하는 시간은 포함되지 않음(wait Queue, I/O수행 시간))
6. Response time – 사용자에게 응답 시간

* CPU-bound process는 a, b에 초점, I/O-bound process는 c, d, e에 초점을 맞추는 것이 좋다.
* CPU-bound process는 context switching을 적게 하는 것이 좋고, I/O-bound process는 많이 하는 것이 좋다. 현재는 두가지를 모두 만족하는 방법은 없다.
* Trade-off의 관계 – 하나가 좋아지면 하나가 나빠짐

1. Scheduling Algorithms

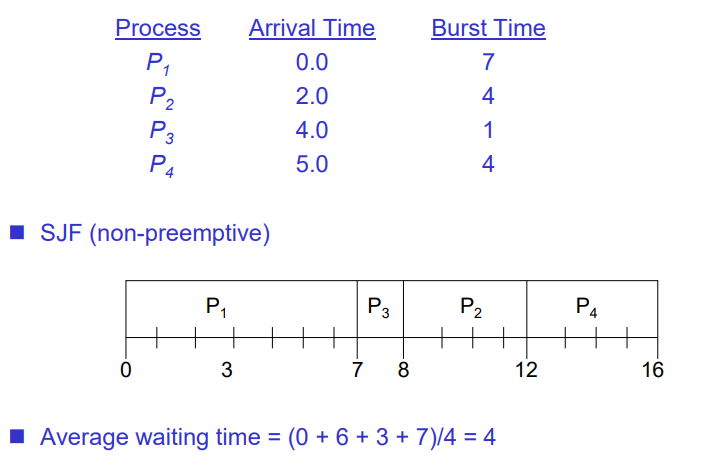
* FCFS Scheduling, SJF Scheduling, Priority Scheduling, Round Robin(RR)

1. First-Come, First-Served(FCFS) Scheduling
   * 먼저 들어온 것을 먼저 수행, Queue의 자료구조(FIFO)
   * 일상생활에서 많이 사용(음식점 줄, 은행, 마트 등)
   * Preemptive
   * 가장 공평한 알고리즘(Fairness)
   * No starvation
   * 평균 Waiting타임 측면에서 아주 좋지 않은 알고리즘
   * Convoy effect – 수행시간이 짧은 프로세스가 뒤로 밀려서 효율성이 매우 떨어지는 현상

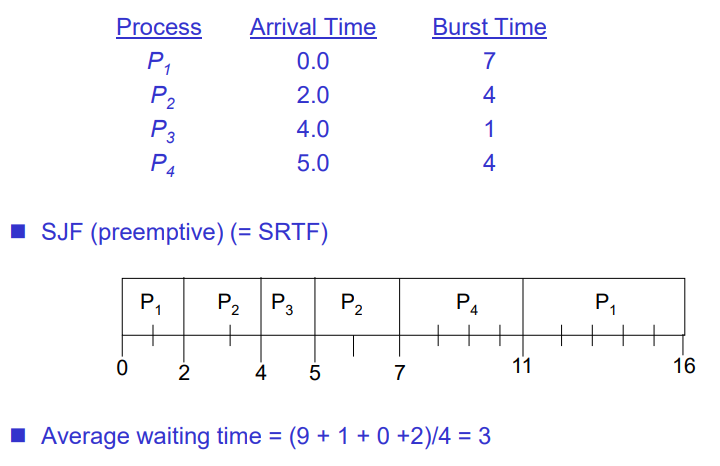


* 어짜피 전체 수행시간에는 변함이 없는데 왜 waiting time을 줄이려고 할까?
* 사용자와 소통해야 하는 interactive 시스템에서는 waiting time을 줄여주는 것이 사용자의 만족도를 높일 수 있기 때문이다.

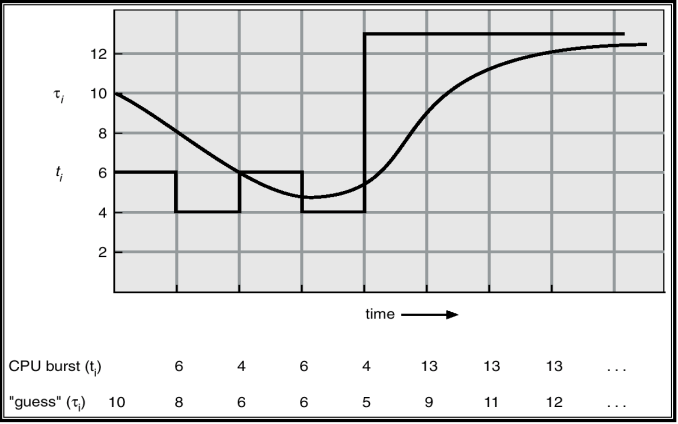
1. Shortest-Job-First(SJF) Scheduling
   * Preemptive VS Non-Preemptive
   * SJF is optimal(waiting 타임 측면)
   * Fair 하지 않음
2. Non-Preemptive SJF



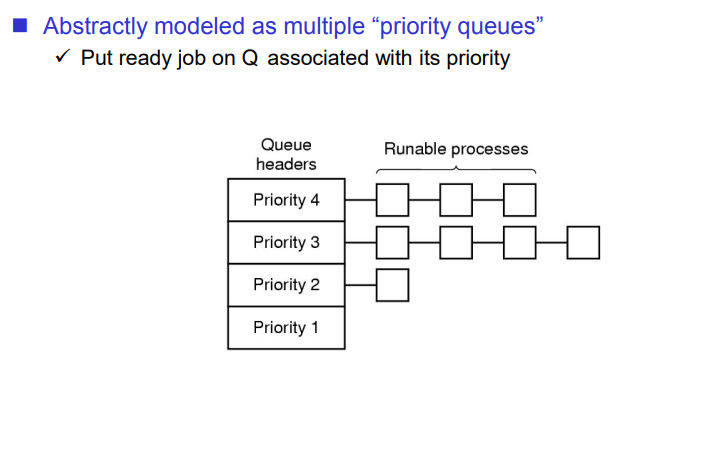
1. Preemptive SJF (=SRTF, Shortest-remaining-time-first)



* + Non-preemtive SJF보다 preemptive SJF이 더 waiting 타임을 줄일 수 있는 방법이다.
  + 하지만 이러한 SJF는 optimal이지만, 실제 컴퓨터에 적용하기엔 무리가 있다. Process를 실제로 실행해 보기 전에는 CPU burst time을 정확히 알 수가 없기 때문.

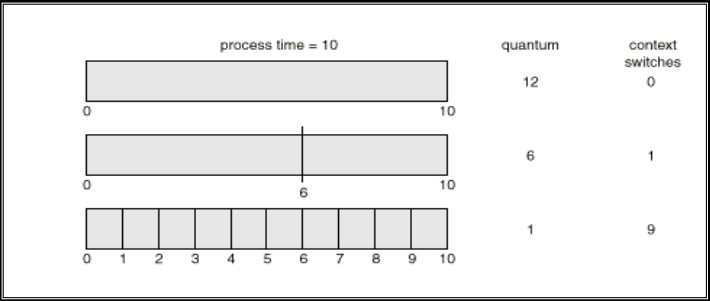


1. Priority Scheduling
   * 우선순위를 줘서 높은 것 먼저 처리
   * Fair하지 않음 -> Problem=starvation현상 -> Solution=Aging

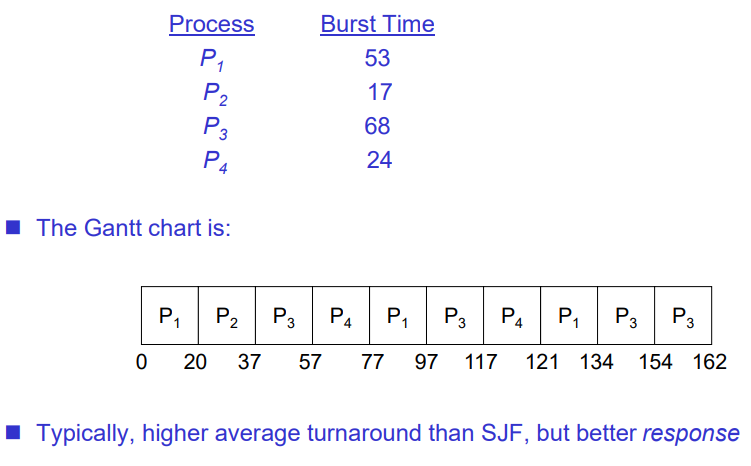


* + Priority가 같은 process들끼리 줄 세우고 높은 순으로 처리
  + process에서 어떤 기준으로 priority를 줄 것인가?
  + 같은 priority안에 있는 process는 어떻게 처리할 것인가?
  + Priority4는 SCFS, priority3는 SJF 이러한 방법으로 설정 가능
  + 각 Priority를 Round Robin(RR)으로 설정하는 것이 가장 합당

1. Round Robin(RR)
   * Time quantum을 두고 돌아가면서 처리하는 방식
   * Robin은 새 이름(새끼들에게 밥을 골고루 나눠서 준다고 함)
   * Time quantum을 어떻게 설정하느냐에 따라 성능에 차이가 있음
   * Time Quentum and Context Switch Time



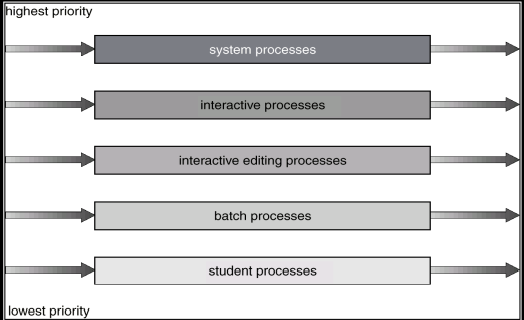
* + 첫번째 막대- CPU-bound process에 유리
  + 세번째 막대- I/O-bound process에 유리
  + Trade-off
  + 적절하게 time quantum을 설정하는 것이 중요
  + CPU burst time(평균?)보다 약간 더 크게 설정하는 것이 일반적
  + 현재 리눅스에서는 60ms의 값을 취하고 있다. 즉 우리가 수행하는 process의 CPU burst time이 60ms보다 조금 작다.
  + Example) time quantum = 20



1. Multilevel Queue(다단계 큐) - Combining Algorithms

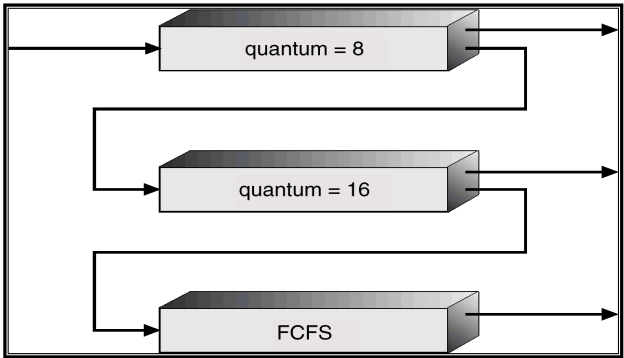
* Queue를 여러 개 두고 각각에 특성에 맞는 알고리즘을 적용하는 방식
* Foreground(interactive) – RR (워드 작업, 화면에 마우스가 느리게 움직이면 화가 남)
* Background(batch) – FCFS (영화 다운)
* foreground queue와 background queue 두개 사이엔 어떤 스케줄링이 적용되어야 할까?

1. Fixed priority scheduling- foreground를 모두 수행 후 background 수행
2. Time slice – ex) 80%는 foreground 수행, 20%는 background 수행



1. Multilevel Feedback Queue

* 기존의 multilevel queue는 프로세스가 큐들 간에 이동이 불가능함. 이것을 보완해서 만든 것이 multilevel feedback queue임



* 위 그림과 같이 우선은 모든 프로세스를 priority가 가장 높은 queue에 집어 넣은 후, time quantum 별로 단계를 만들어서 걸러낸다. (필터 같은 역할)
* Ex) 영화 다운로드시 CPU burst time이 길기 때문에 자연스럽게 FCFS 로 걸러지고 마우스 사용이나 키보드 사용과 같은 interactive process는 작업량이 적은 I/O bound process이기 때문에 quantum=8에서 걸러진다.