Chapter 6

프로세스 스케줄링

Process Scheduling

다중프로그래밍 (Multi-programming)

- **여러개의 프로세스가 시스템 내 존재**
- **자원을 할당 할 프로세스를 선택 해야 함**
  - **스케줄링(Scheduling)**
- **자원 관리**
  - **시간 분할 (time sharing) 관리**
    - 하나의 자원을 여러 스레드들이 번갈아 가며 사용
    - 예) 프로세서 (Processor)
    - **프로세스 스케줄링 (Process scheduling)**
      - 프로세서 사용시간을 프로세스들에게 분배
  - **공간 분할 (space sharing) 관리**
    - 하나의 자원을 분할하여 동시에 사용
    - 예) 메모리 (memory)

스케줄링(Scheduling)의 목적

- **시스템의 성능(performance) 향상**

- **대표적 시스템 성능 지표 (index)**
  - **응답시간 (response time)**
    - 작업 요청(submission)으로부터 응답을 받을때까지의 시간
  - **작업 처리량 (throughput)**
    - 단위 시간 동안 완료된 작업의 수
  - **자원 활용도 (resource utilization)**
    - 주어진 시간(T_c) 동안 자원이 활용된 시간(T_r)

$$Utilization = \frac{T_r}{T_c}$$

- **목적에 맞는 지표를 고려하여 스케줄링 기법을 선택**

시스템 성능 지표들

- **평균 응답 시간 (mean response time)**
  - 사용자 지향적, 예) interactive systems
- **처리량 (throughput)**
  - 시스템 지향적, 예) batch systems
- **자원 활용도 (resource utilization)**
- **공평성(fairness)**
  - 예) FIFO
- **실행 대기 방지**
  - 무기한 대기 방지
- **예측 가능성(predictability)**
  - 적절한 시간안에 응답을 보장하는가
- **…**

시스템 성능 지표들 (in textbook)

- 자원 할당의 공정성 보장
- 단위시간당 처리량 최대화
- 적절한 반환시간 보장
- 예측 가능성 보장
- 오버헤드 최소화
- 자원 사용의 균형 유지
- 반환시간과 자원의 활용 간에 균형 유지
- 실행 대기 방지
- 우선순위
- 서비스 사용 기회 확대
- 서비스 수 감소 방지

대기시간, 응답시간, 반환시간

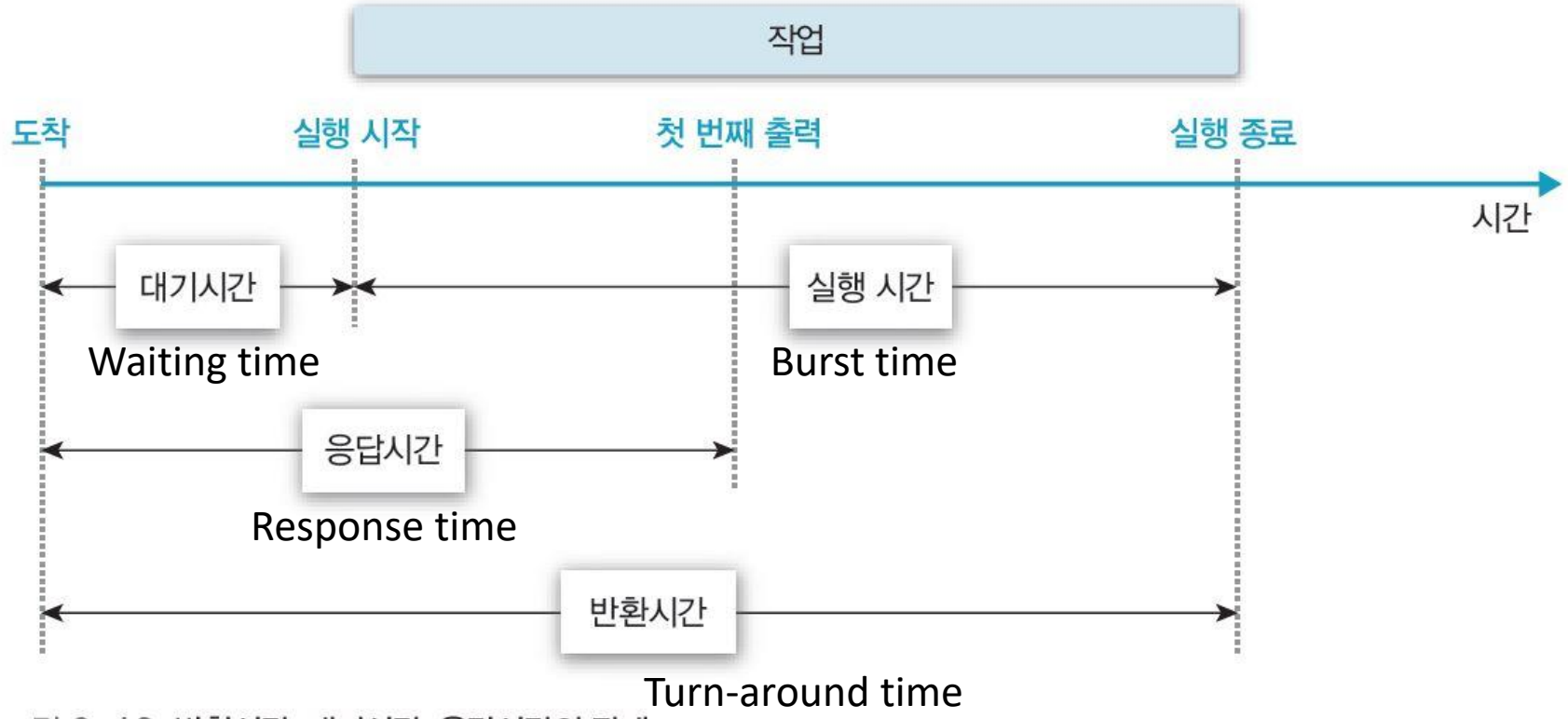


그림 6-13 반환시간, 대기시간, 응답시간의 관계

개요

- 스케줄링의 목적
- **스케줄링 기준 및 단계**
- **스케줄링 정책**
- **기본 스케줄링 알고리즘들**
- **Case study**

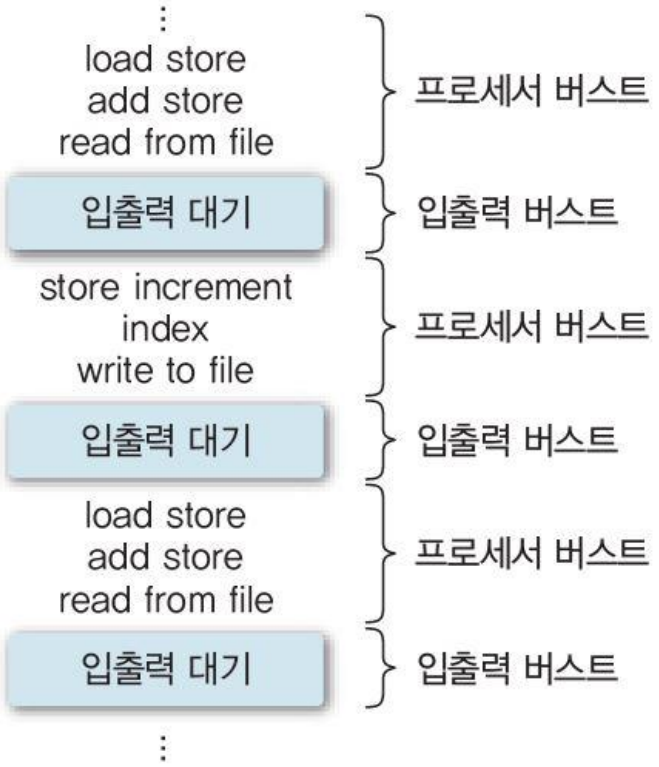
스케줄링 기준 (Criteria)

- **스케줄링 기법이 고려하는 항목들**

- **프로세스(process)의 특성**
  - I/O-bounded or compute-bounded
- **시스템 특성**
  - Batch system or interactive system
- **프로세스의 긴급성(urgency)**
  - Hard- or soft- real time, non-real time systems
- **프로세스 우선순위 (priority)**
- **프로세스 총 실행 시간 (total service time)**
- **…**

CPU burst vs I/O burst

- **프로세스 수행**
  = CPU 사용 + I/O 대기
- **CPU burst**
  - CPU 사용 시간
- **I/O burst**
  - I/O 대기 시간
- **Burst time은 스케줄링의 중요한 기준 중 하나**



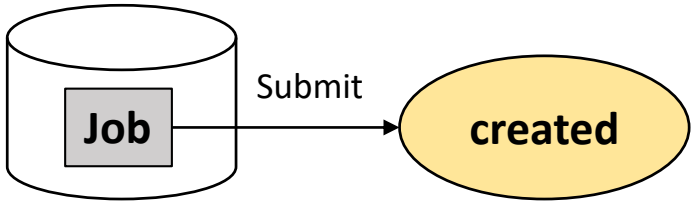
스케줄링의 단계 (Level)

- **발생하는 빈도 및 할당 자원에 따른 구분**

- **Long-term scheduling**
  - 장기 스케줄링
  - Job scheduling
- **Mid-term scheduling**
  - 중기 스케줄링
  - Memory allocation
- **Short-term scheduling**
  - 단기 스케줄링
  - Process scheduling

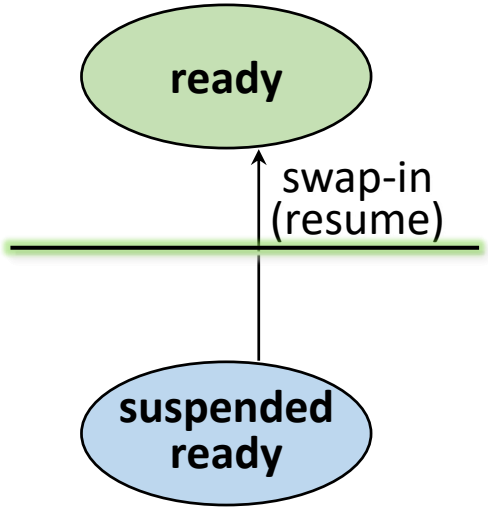
Long-term Scheduling

- **Job scheduling**
  - 시스템에 제출 할 (Kernel에 등록 할) 작업(Job) 결정
    - Admission scheduling, High-level scheduling
- **다중프로그래밍 정도(degree) 조절**
  - 시스템 내에 프로세스 수 조절
- **I/O-bounded 와 compute-bounded 프로세스들을 잘 섞어서 선택해야 함**
  - **why?**
- **시분할 시스템에서는 모든 작업을 시스템에 등록**
  - Long-term scheduling이 불필요



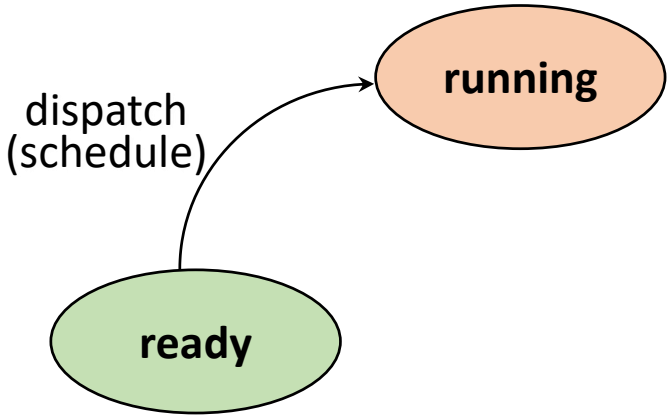
Mid-term Scheduling

- **메모리 할당 결정 (memory allocation)**
  - Intermediate-level scheduling
  - Swapping (swap-in/swap-out)

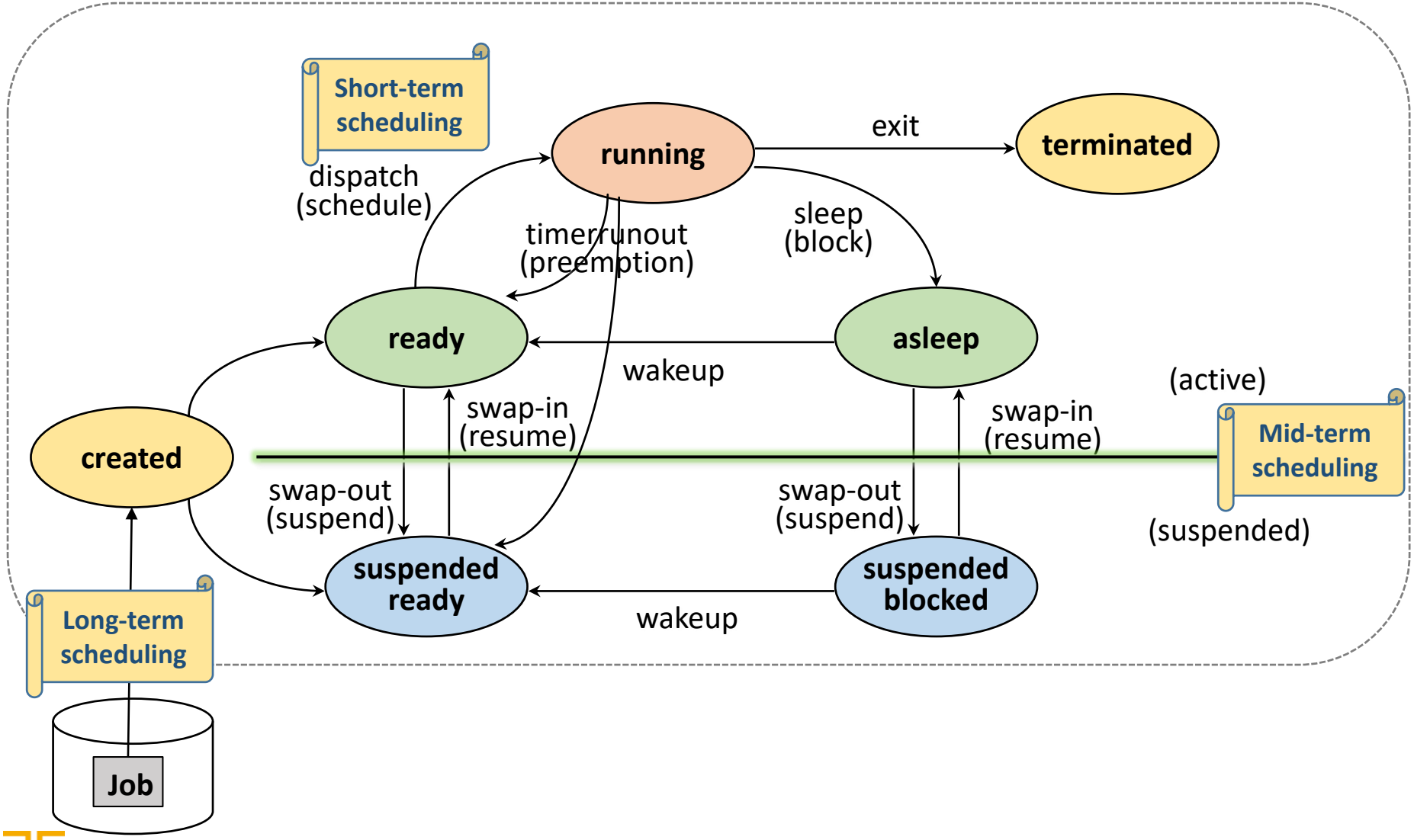


Short-term Scheduling

- **Process scheduling**
  - Low-level scheduling
  - 프로세서(processor)를 할당할 프로세스(process)를 결정
    - Processor scheduler, dispatcher
- **가장 빈번하게 발생**
  - Interrupt, block (I/O), time-out, Etc.
  - 매우 빨라야 함
    - E.g.,
    - average CPU burst = 100ms
      scheduling decision = 10ms
    - 10 x (100+10) = 9%
      of the CPU is being used simply for scheduling



스케줄링의 단계 (Level)



개요

- 스케줄링의 목적
- 스케줄링 기준 및 단계
- **스케줄링 정책**
- **기본 스케줄링 알고리즘들**
- **Case study**

스케줄링 정책 (Policy)

- **선점 vs 비선점**
  - Preemptive scheduling, Non-preemptive scheduling
- **우선순위**
  - Priority

Preemptive/Non-preemptive scheduling

- **Non-preemptive scheduling**
  - 할당 받을 자원을 스스로 반납할 때까지 사용
    - 예) system call, I/O, Etc.
  - 장점
    - Context switch overhead가 적음
  - 단점
    - 잦은 우선순위 역전, 평균 응답 시간 증가

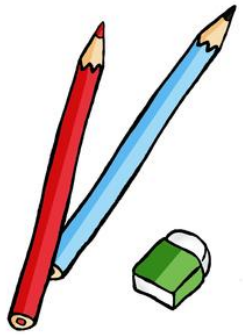
- **Preemptive scheduling**
  - 타의에 의해 자원을 빼앗길 수 있음
    - 예) 할당 시간 종료, 우선순위가 높은 프로세스 등장
  - Context switch overhead가 큼
  - Time-sharing system, real-time system 등에 적합

Priority

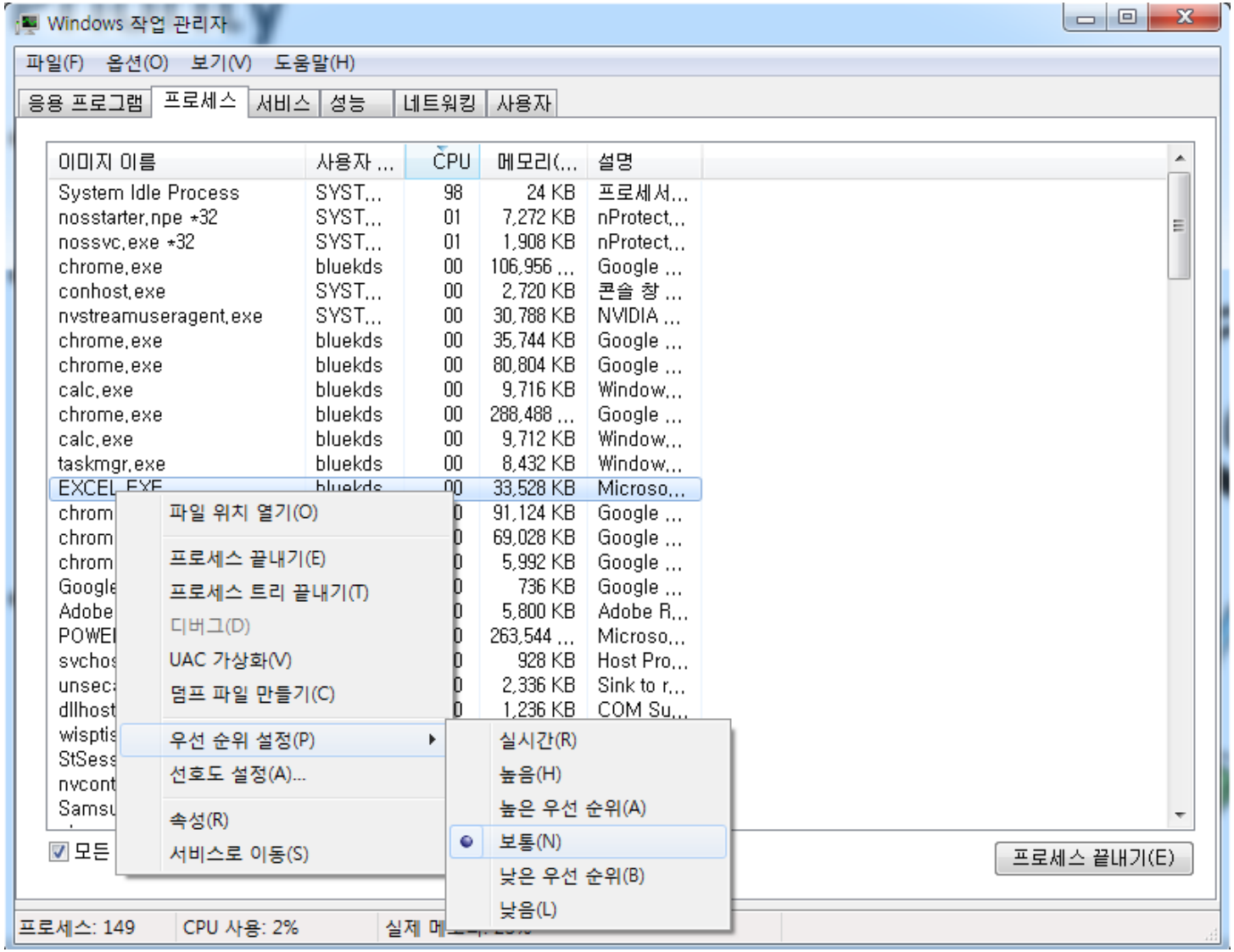
- **프로세스의 중요도**

- **Static priority (정적 우선순위)**
  - 프로세스 생성시 결정된 priority가 유지 됨
  - 구현이 쉽고, overhead가 적음
  - 시스템 환경 변화에 대한 대응이 어려움

- **Dynamic priority (동적 우선순위)**
  - 프로세스의 상태 변화에 따라 priority 변경
  - 구현이 복잡, priority 재계산 overhead가 큼
  - 시스템 환경 변화에 유연한 대응 가능



Priority


Windows 작업 관리자
파일(F) 옵션(O) 보기(V) 도움말(H)
응용 프로그램 프로세스 서비스 성능 네트워킹 사용자
이미지 이름 사용자 ... CPU 메모리(... 설명
System Idle Process SYST... 98 24 KB 프로세서...
nosstarter.npe *32 SYST... 01 7,272 KB nProtect...
nossvc.exe *32 SYST... 01 1,908 KB nProtect...
chrome.exe bluekds 00 106,956 ... Google ...
conhost.exe SYST... 00 2,720 KB 콘솔 창 ...
nvstreamuseragent.exe SYST... 00 30,788 KB NVIDIA ...
chrome.exe bluekds 00 35,744 KB Google ...
chrome.exe bluekds 00 80,804 KB Google ...
calc.exe bluekds 00 9,716 KB Window...
chrome.exe bluekds 00 288,488 ... Google ...
calc.exe bluekds 00 9,712 KB Window...
taskmgr.exe bluekds 00 8,432 KB Window...
EXCEL.EXE 33,528 KB Microso...
파일 위치 열기(O)
프로세스 끝내기(E)
프로세스 트리 끝내기(T)
디버그(D)
UAC 가상화(V)
덤프 파일 만들기(C)
우선 순위 설정(P)
선호도 설정(A)...
속성(R)
서비스로 이동(S)
실시간(R)
높음(H)
높은 우선 순위(A)
보통(N)
낮은 우선 순위(B)
낮음(L)
모든
프로세스 끝내기(E)
프로세스: 149
CPU 사용: 2%

요약: Scheduling Concepts

- **멀티프로그래밍 (멀티테스킹)**
- **스케줄링 개념**
  - 목적
  - 성능 지표 (index)
    - CPU burst VS I/O burst
  - 스케줄링 기준(Criteria)
- **스케줄링 레벨**
  - Long-term, Mid-term, Short-term
- **스케줄링 정책**
  - Preemptive/non-preemptive
  - Priority

개요

- 스케줄링의 목적
- 스케줄링 기준 및 단계
- 스케줄링 정책
- **기본 스케줄링 알고리즘들**
- **Case study**

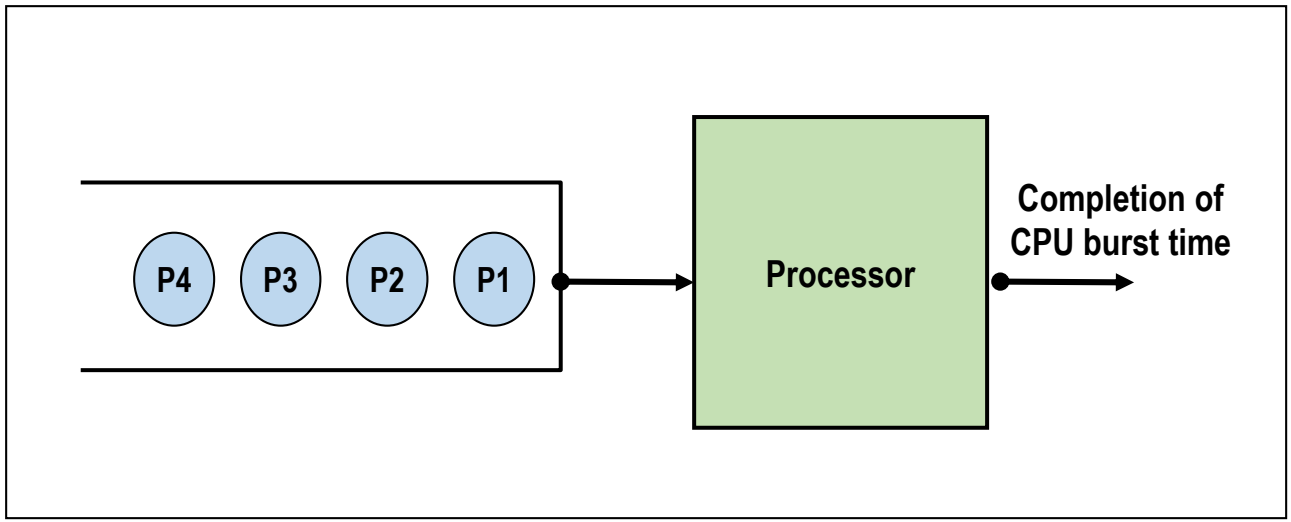
Basic Scheduling algorithms

- FCFS (First-Come-First-Service)
- RR (Round-Robin)
- SPN (Shortest-Process-Next)
- SRTN (Shortest Remaining Time Next)
- HRRN (High-Response-Ratio-Next)
- MLQ (Multi-level Queue)
- MFQ (Multi-level Feedback Queue)

FCFS (First-Come-First-Service)

- **Non-preemptive scheduling**
- **스케줄링 기준 (Criteria)**
  - **도착 시간** (ready queue 기준)
  - 먼저 도착한 프로세스를 먼저 처리
- **자원을 효율적으로 사용 가능**
  - High resource utilization / **why?**
- **Batch system에 적합, interactive system에 부적합**
- **단점**
  - **Convoy effect**
    - 하나의 수행시간이 긴 프로세스에 의해 다른 프로세스들이 긴 대기시간을 갖게 되는 현상 (대기시간 >> 실행 시간)
  - **긴 평균 응답시간(response time)**

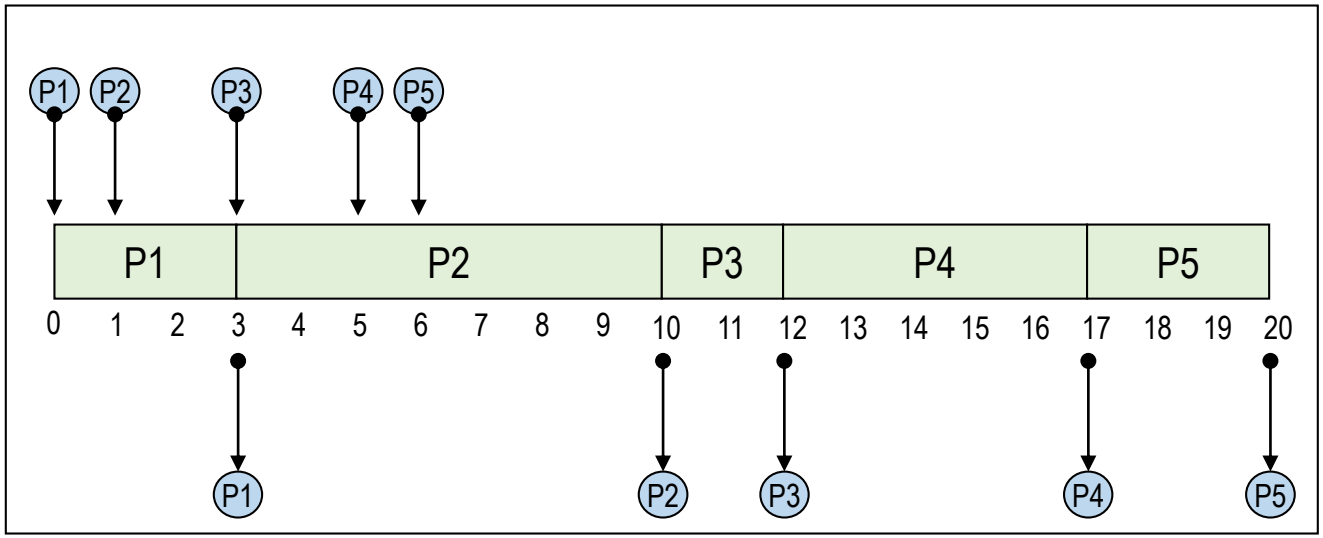
FCFS (First-Come-First-Service)



Base images from Prof. Seo's slides

FCFS (First-Come-First-Service)

Base images from Prof. Seo's slides



Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3	0	3	3/3 = 1
P2	1	7	2	9	9/7 = 1.3
P3	3	2	7	9	9/2 = 4.5
P4	5	5	7	12	12/5 = 2.4
P5	6	3	11	14	14/3 = 4.7

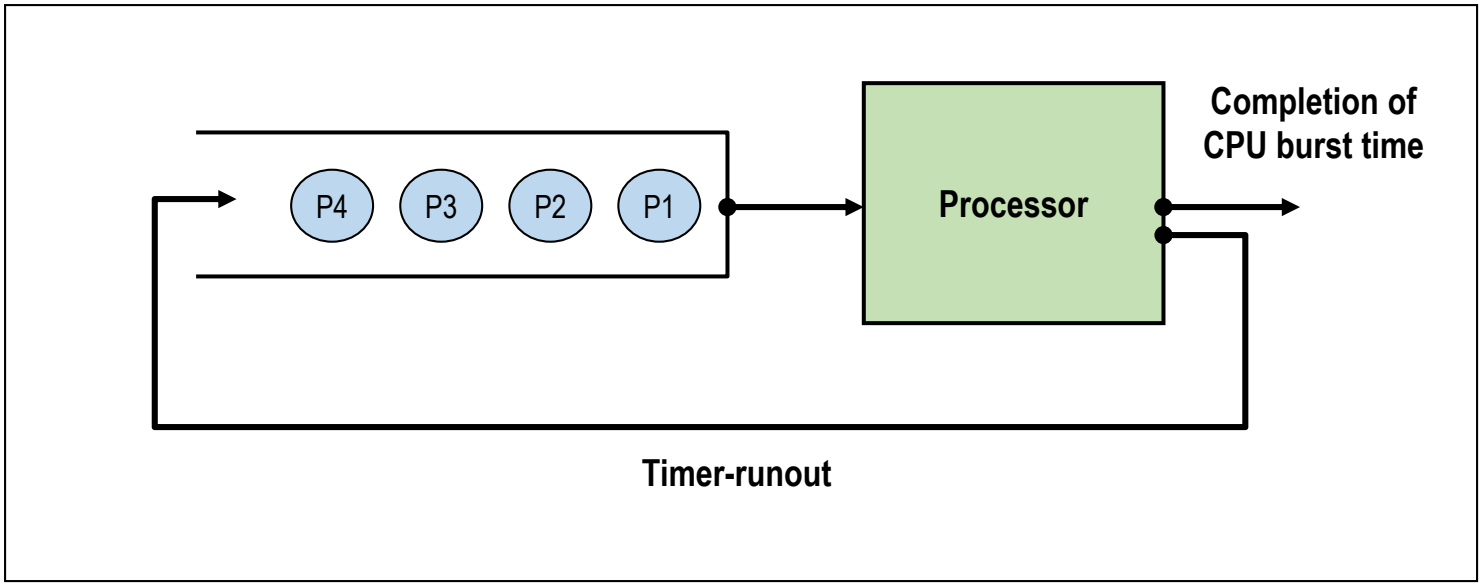
RR (Round-Robin)

- **Preemptive scheduling**
- **스케줄링 기준 (Criteria)**
  - **도착 시간** (ready queue 기준)
  - 먼저 도착한 프로세스를 먼저 처리
- **자원 사용 제한 시간(time quantum)이 있음**
  - System parameter (δ)
  - 프로세스는 할당된 시간이 지나면 자원 반납
    - Timer-runout
  - 특정 프로세스의 자원 독점(monopoly) 방지
  - Context switch overhead가 큼
- **대화형, 시분할 시스템에 적합**

RR (Round-Robin)

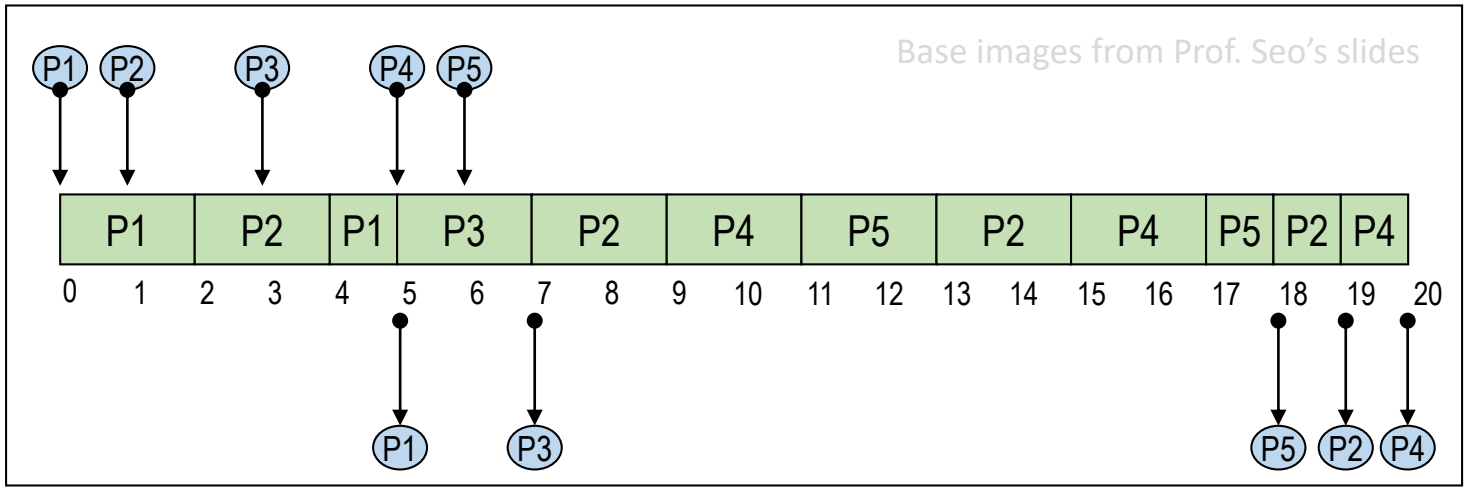
- **Time quantum (δ)이 시스템 성능을 결정하는 핵심 요소**
  - Very large (infinite) δ → FCFS
  - Very small time quantum → processor sharing
    - 사용자는 모든 프로세스가 각각의 프로세서 위에서 실행되는 것처럼 느낌
      - 체감 프로세서 속도 = 실제 프로세서 성능의 1/n
    - High context switch overhead

RR (Round-Robin)



Base images from Prof. Seo's slides

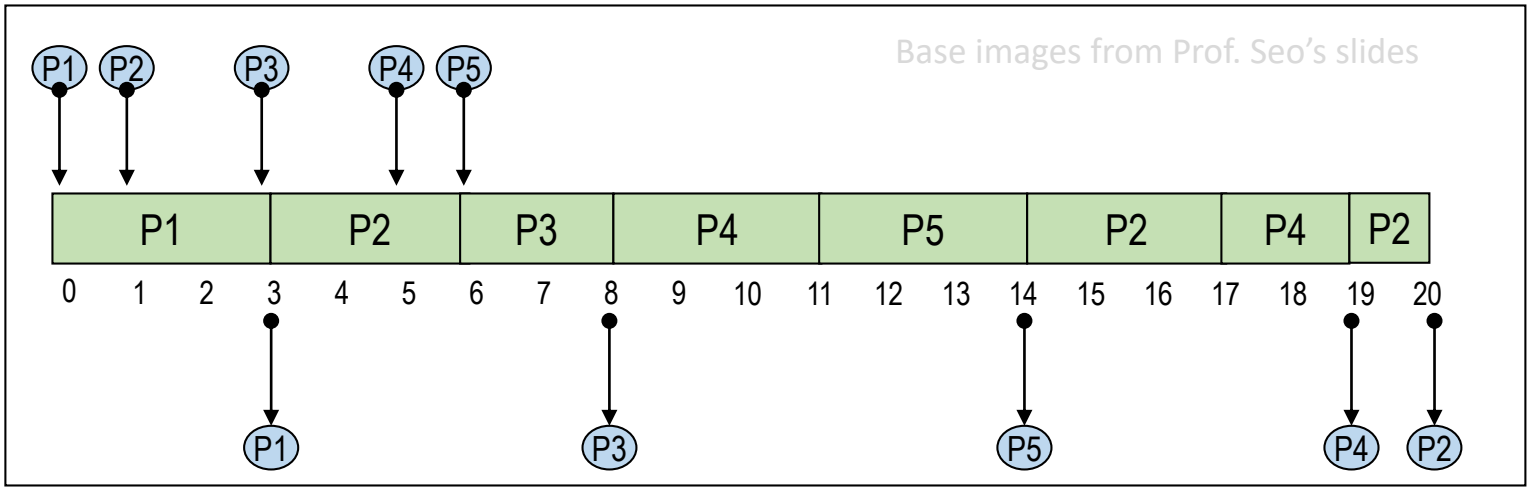
RR (Round-Robin), δ = 2



Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3	2	5	5/3 = 1.7
P2	1	7	11	18	18/7 = 2.6
P3	3	2	2	4	4/2 = 2
P4	5	5	10	15	15/5 = 3
P5	6	3	9	12	12/3 = 4

Average response time(TT) =

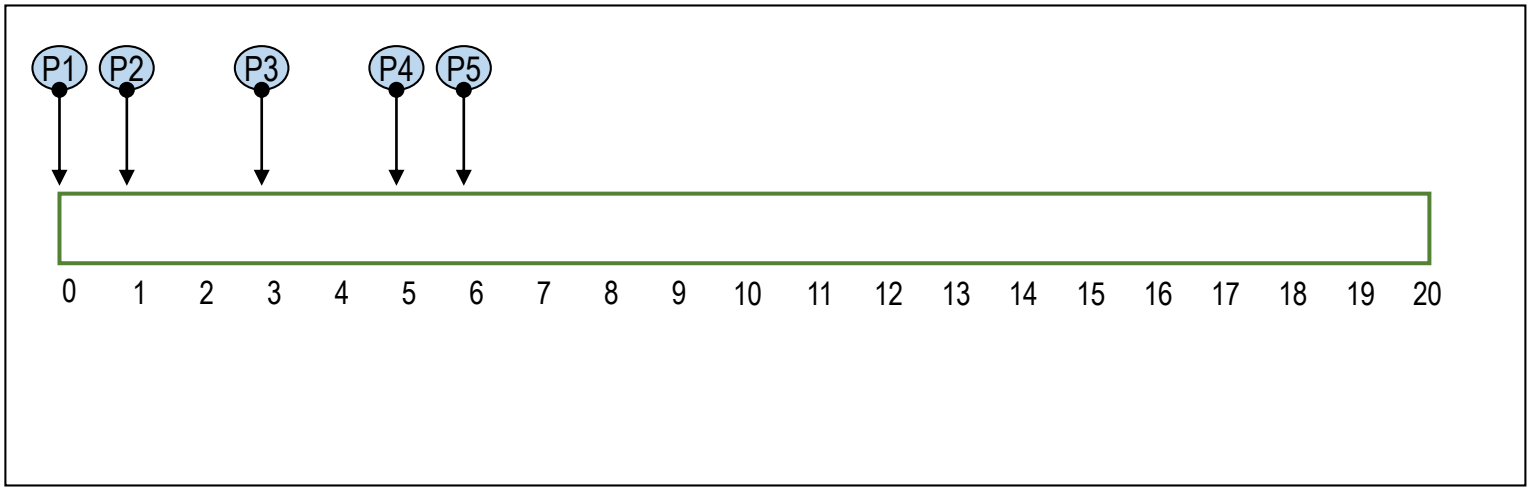
RR (Round-Robin), δ = 3



Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3			
P2	1	7			
P3	3	2			
P4	5	5			
P5	6	3			

Average response time(TT) =

RR (Round-Robin), δ = 4

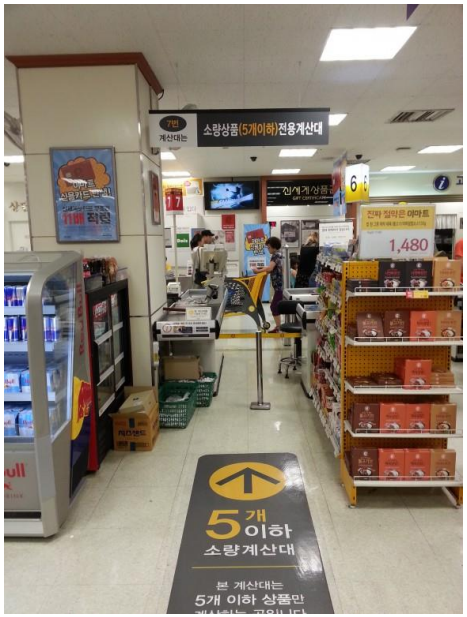


Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3			
P2	1	7			
P3	3	2			
P4	5	5			
P5	6	3			

Average response time(TT) =

SPN (Shortest-Process-Next)

- **Non-preemptive scheduling**
- **스케줄링 기준 (Criteria)**
  - **실행시간** (burst time 기준)
  - Burst time 가장 작은 프로세스를 먼저 처리
    - SJF(Shortest Job First) scheduling



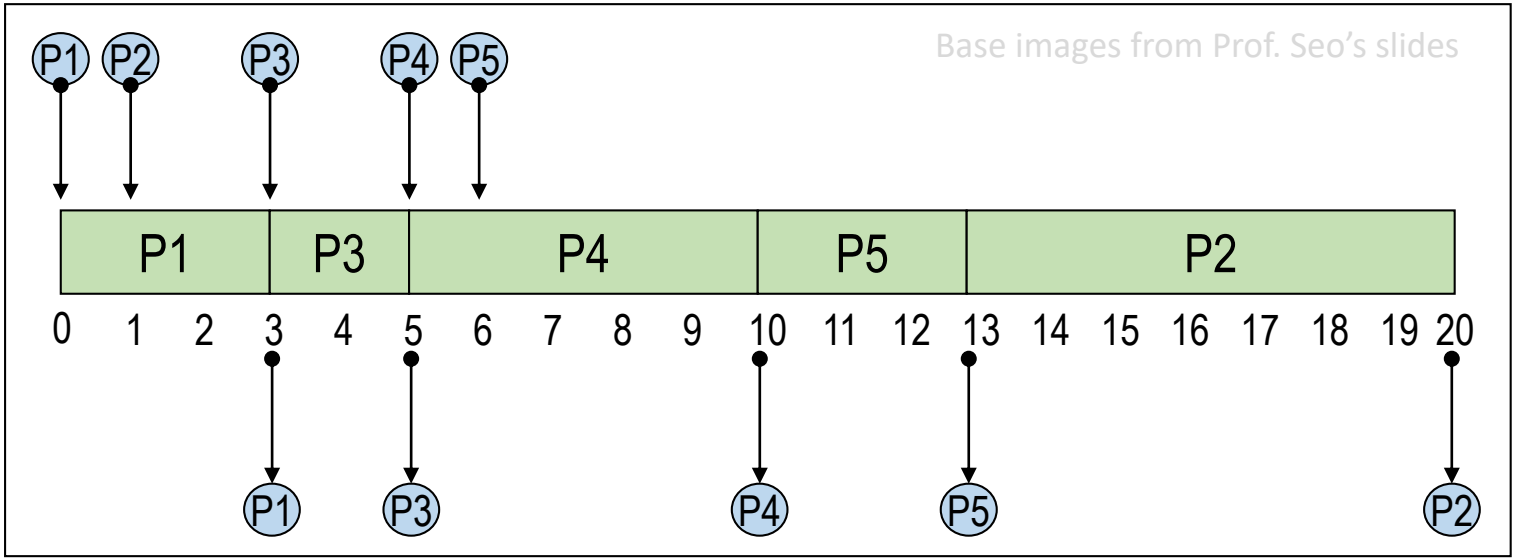
SPN (Shortest-Process-Next)

- **장점**
  - 평균 대기시간(WT) 최소화
  - 시스템 내 프로세스 수 최소화
    - 스케줄링 부하 감소, 메모리 절약 → 시스템 효율 향상
  - 많은 프로세스들에게 빠른 응답 시간 제공

- **단점**
  - **Starvation** (무한대기) 현상 발생
    - BT가 긴 프로세스는 자원을 할당 받지 못 할 수 있음
      - Aging 등으로 해결 (e.g., HRRN)
  - 정확한 실행시간을 알 수 없음
    - 실행시간 예측 기법이 필요

Image from
https://www.toonpool.com/

SPN (Shortest-Process-Next)



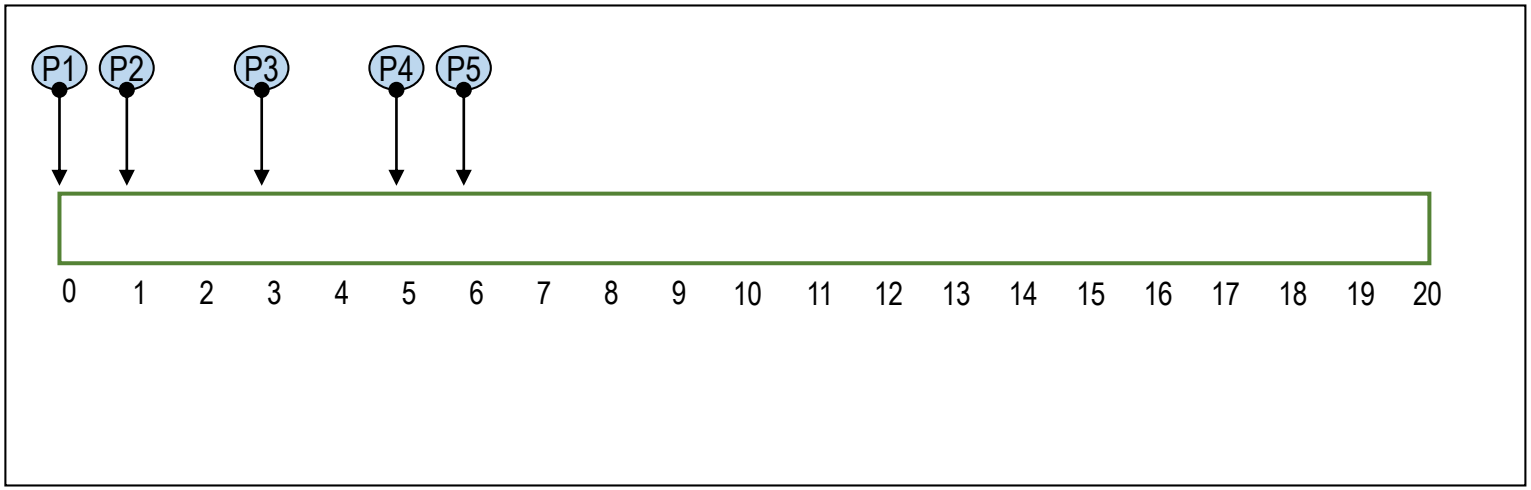
Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3	0	3	3/3 = 1
P2	1	7	12	19	19/7 = 2.7
P3	3	2	0	2	2/2 = 1
P4	5	5	0	5	5/5 = 1
P5	6	3	4	7	7/3 = 2.3

SRTN (Shortest Remaining Time Next)

- **SPN의 변형**
- **Preemptive scheduling**
  - 잔여 실행 시간이 더 적은 프로세스가 ready 상태가 되면 선점됨
- **장점**
  - SPN의 장점 극대화
- **단점**
  - 프로세스 생성시, 총 실행 시간 예측이 필요함
  - 잔여 실행을 계속 추적해야 함 = overhead
  - Context switching overhead

→ 구현 및 사용이 비현실적

SRTN (Shortest Remaining Time Next)

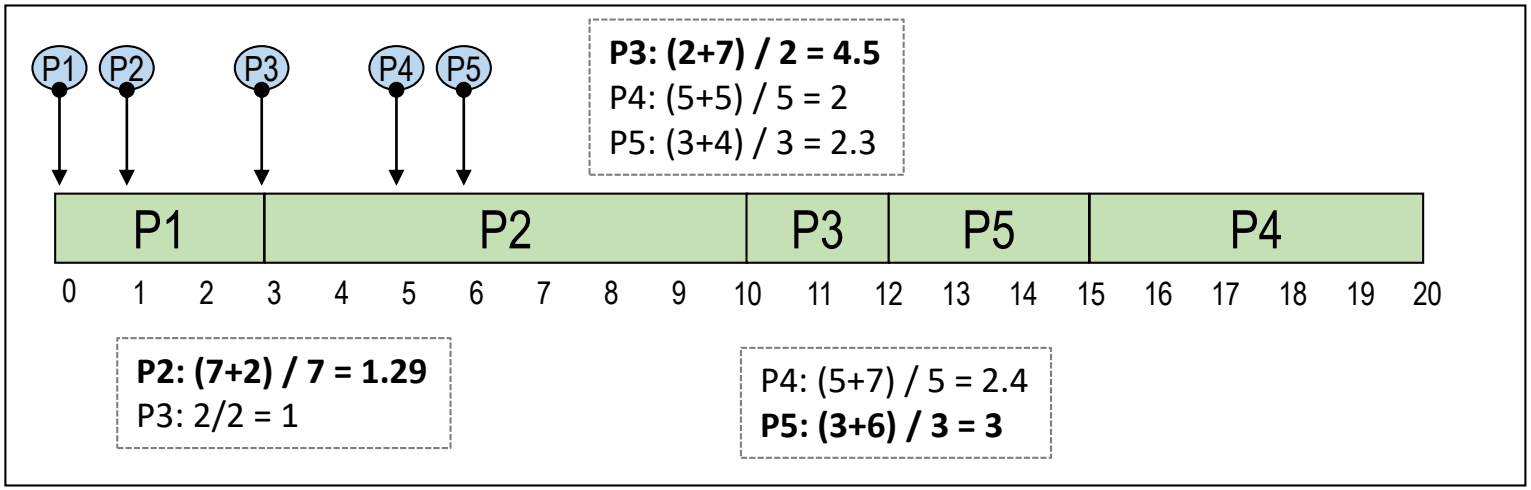


Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3			
P2	1	7			
P3	3	2			
P4	5	5			
P5	6	3			

HRRN (High-Response-Ratio-Next)

- **SPN의 변형**
  - SPN + Aging concepts, Non-preemptive scheduling
- **Aging concepts**
  - 프로세스의 대기 시간(WT)을 고려하여 기회를 제공
- **스케줄링 기준 (Criteria)**
  - Response ratio가 높은 프로세스 우선
- $\boldsymbol{Response\ ratio = \frac{WT+BT}{BT}}$ **(응답률)**
  - SPN의 장점 + Starvation 방지
  - 실행 시간 예측 기법 필요 (overhead)

HRRN (High-Response-Ratio-Next)



Process ID	Arrival time	Burst time (BT)	Waiting time (WT) = TT-BT	Turnaround time (TT)	Normalized TT (NTT) = TT/BT
P1	0	3	0	3	3/3 = 1
P2	1	7	2	9	9/7 = 1.29
P3	3	2	7	9	9/2 = 4.5
P4	5	5	10	15	15/5 = 3
P5	6	3	6	9	9/3 = 3

Basic Scheduling algorithms

- **FCFS (First-Come-First-Service)**
- **RR (Round-Robin)**

Fairness
(공평성)

- **SPN (Shortest-Process-Next)**
- **SRTN (Shortest Remaining Time Next)**
- **HRRN (High-Response-Ratio-Next)**

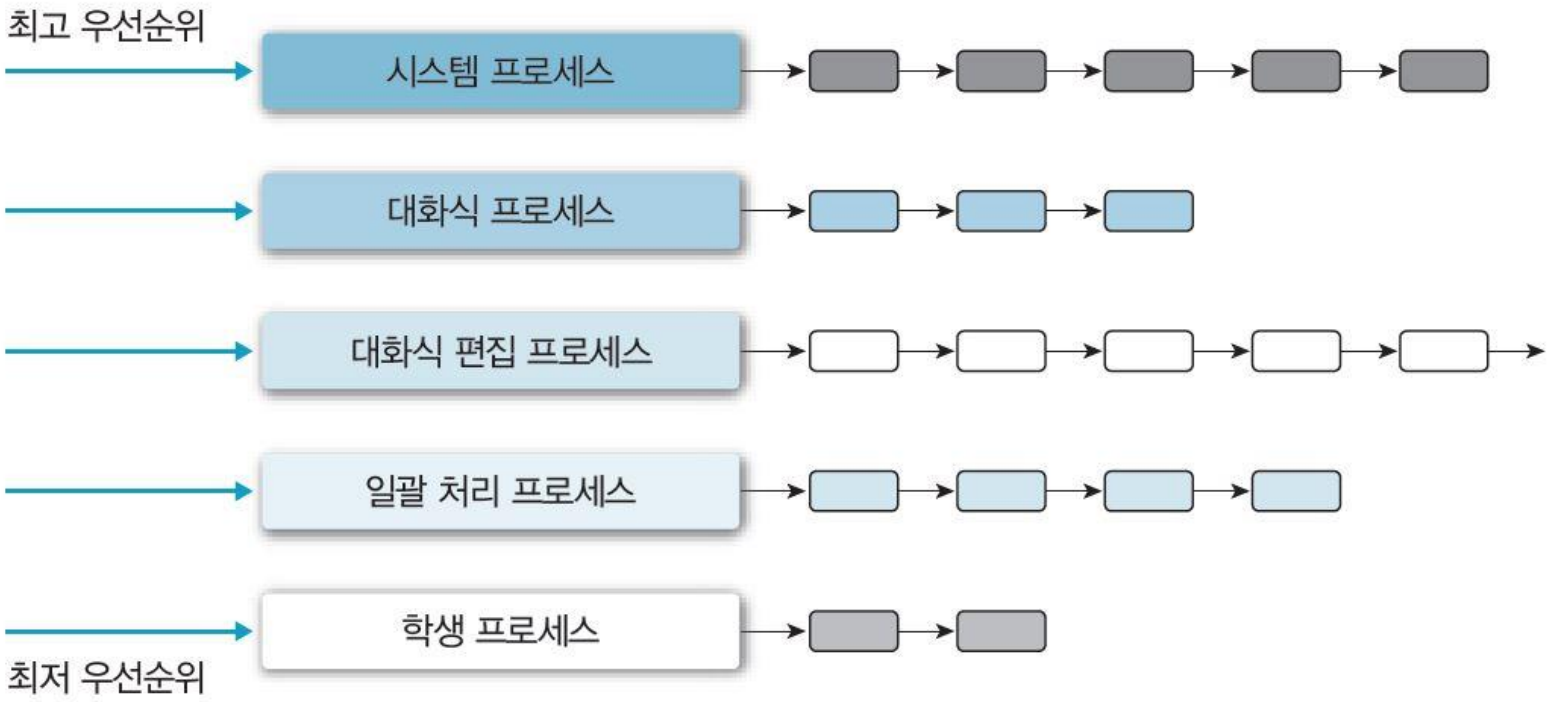
Efficiency/Performance
(효율성, 성능)
- 문제점
  - 실행시간 예측 부하

- **MLQ (Multi-level Queue)**
- **MFQ (Multi-level Feedback Queue)**

MLQ (Multi-level Queue)

- **작업 (or 우선순위)별 별도의 ready queue를 가짐**
  - 최초 배정 된 queue를 벗어나지 못함
  - 각각의 queue는 자신만의 스케줄링 기법 사용
- **Queue 사이에는 우선순위 기반의 스케줄링 사용**
  - E.g., fixed-priority preemptive scheduling
- **장점**
  - 빠른 응답시간 (?)
- **단점**
  - 여러 개의 Queue 관리 등 스케줄링 overhead
  - 우선순위가 낮은 queue는 starvation 현상 발생 가능

MLQ (Multi-level Queue)

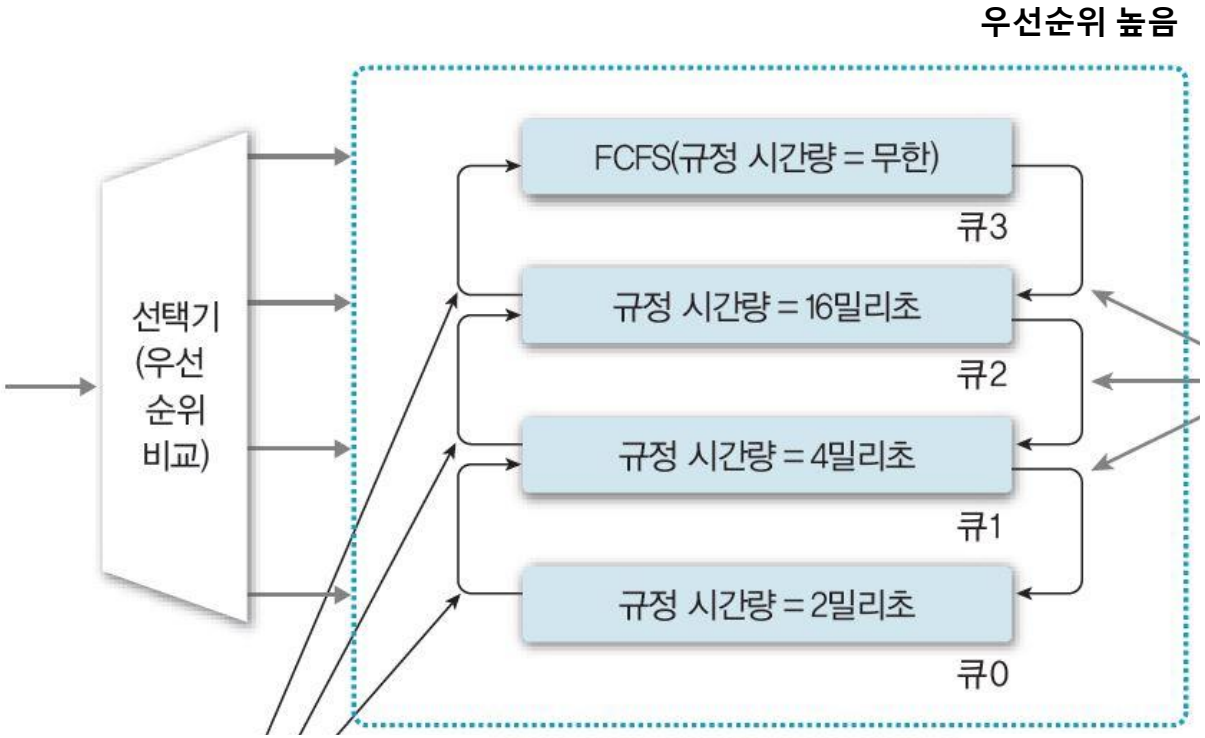


Queue의 구성은 정책에 따라 결정

MFQ (Multi-level Feedback Queue)

- **프로세스의 Queue간 이동이 허용된 MLQ**
- **Feedback을 통해 우선 순위 조정**
  - 현재까지의 프로세서 사용 정보(패턴) 활용
- **특성**
  - Dynamic priority
  - Preemptive scheduling
  - Favor short burst-time processes
  - Favor I/O bounded processes
  - Improve adaptability
- **프로세스에 대한 사전 정보 없이 SPN, SRTN, HRRN 기법의 효과를 볼 수 있음**

MFQ (Multi-level Feedback Queue)



- Computed-bounded
- …

- Aging
- I/O-bounded
- …

MFQ (Multi-level Feedback Queue)

- **단점**
  - 설계 및 구현이 복잡, 스케줄링 overhead가 큼
  - Starvation 문제 등

- **변형**
  - **각 준비 큐마다 시간 할당량을 다르게 배정**
    - 프로세스의 특성에 맞는 형태로 시스템 운영 가능
  - **입출력 위주 프로세스들을 상위 단계의 큐로 이동, 우선 순위 높임**
    - 프로세스가 block될 때 상위의 준비 큐로 진입하게 함
    - 시스템 전체의 평균 응답 시간 줄임, 입출력 작업 분산 시킴
  - **대기 시간이 지정된 시간을 초과한 프로세스들을 상위 큐로 이동**
    - 에이징 (aging) 기법

MFQ (Multi-level Feedback Queue)

- **Parameters for MFQ scheduling**
  - Queue의 수
  - Queue별 스케줄링 알고리즘
  - 우선 순위 조정 기준
  - 최초 Queue 배정 방법
  - …

요약: Basic Scheduling algorithms

- FCFS (First-Come-First-Service)
- RR (Round-Robin)

- SPN (Shortest-Process-Next)
- SRTN (Shortest Remaining Time Next)
- HRRN (High-Response-Ratio-Next)

- MLQ (Multi-level Queue)
- MFQ (Multi-level Feedback Queue)

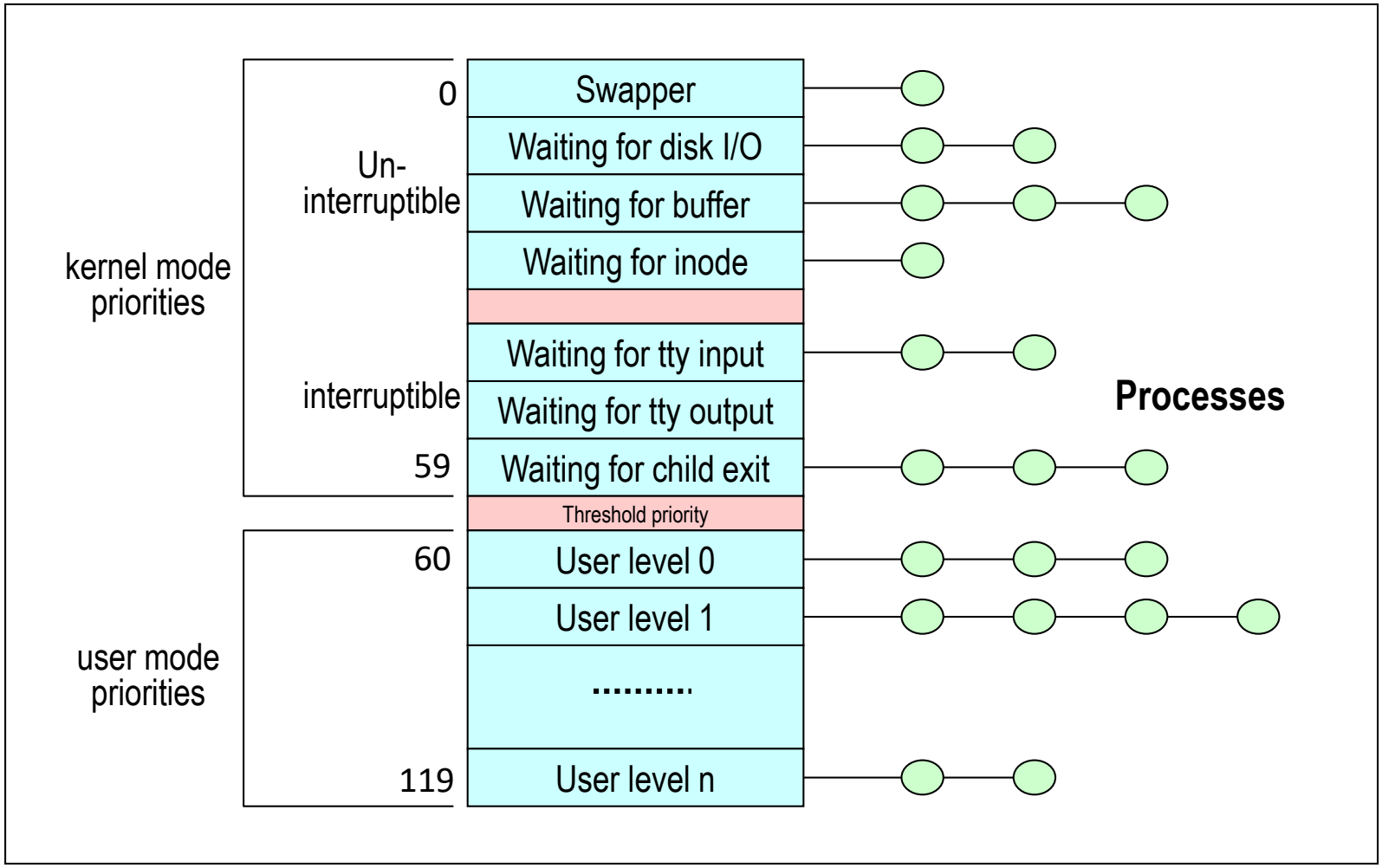
Case study: Scheduling in OSs

- **Scheduling in Unix**
- **Scheduling in Windows**

Scheduling in Unix Systems

- **Interactive system**
  - Priority-based scheduling, preemptive scheduling
- **Priority**
  - Kernel priority
    - 커널 모드에 있는 프로세스
    - interruptible/uninterruptible priority
  - User priority
    - 사용자 모드에 있는 프로세스
- **Clock handler**
  - 주기적으로 인터럽트 발생
  - 모든 프로세스들의 우선순위 조정

Scheduling in Unix Systems



Base images from Prof. Seo's slides

Scheduling in Unix Systems

- **스케줄링 기법**
  - 높은 우선 순위 우선
    - 프로세서 사용량에 따라 우선순위가 주기적으로 변함
  - MFQ 스케줄링 기법
- **우선순위 조정**
  - 프로세서 사용 반영 (decay computation)
    - $\boldsymbol{CPUCount} = \frac{\boldsymbol{CPUCount}}{\boldsymbol{2}}$
  - 우선순위 조정
    - $\boldsymbol{Priority} = \frac{\boldsymbol{CPUCount}}{\boldsymbol{2}} + \boldsymbol{basePriority} + \boldsymbol{niceValue}$

https://en.wikipedia.org/wiki/Nice_(Unix)

Scheduling in Unix Systems

	프로세스 P1		프로세스 P1		프로세스 P1	
	우선 순위	CPUCount	우선 순위	CPUCount	우선 순위	CPUCount
0	60	0 1 2 ••• 60 (CPU clock tick)	60	0	60	0
1	75	30	60	0 1 2 ••• 60	60	0
2	67	15	75	30	60	0 1 2 ••• 60
3	63	7 8 9 ••• 67	67	15	75	30
4	76	33	63	7 8 9 ••• 67	67	15
5	68	16	76	33	63	7

Base images from Prof. Seo's slides

Scheduling in Windows

- **Thread scheduling**
- **Priority-based, Preemptive scheduling**
- **32-level priority scheme**
  - Variable class: 1-15
  - Real-time class: 16-32
- **MLQ scheduling**
  - 각각의 우선순위 마다 큐(queue)를 가짐
  - 스케줄러(scheduler)는 높은 우선순위 큐부터 순차적으로 방문하며, 실행할 준비가 된 스레드를 찾음

Scheduling in Windows

- **Priorities in Windows**

Priority class / Priority	real-time	high	above normal	normal	below normal	idle
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Base priority

Variable classes

Scheduling in Windows

- **스케줄링 기법**
  - 스레드가 자신의 time quantum을 다 소비하고 나올때
    → 우선순위 하향 조정 (variable class의 경우)

  - Wake-up 시 → 우선순위 상향 조정 (variable class의 경우)
    - 상향 정도는 스레드가 대기 중이었단 작업에 종속적
      - 예) 키보드/마우스 I/O → 크게 상향
        디스크 I/O → 약간 상향