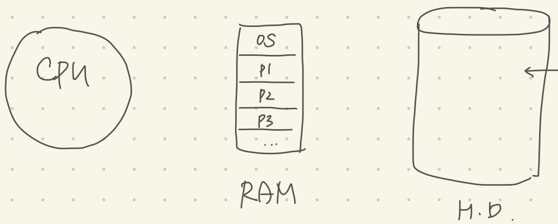
Types of scheduler, Context switching

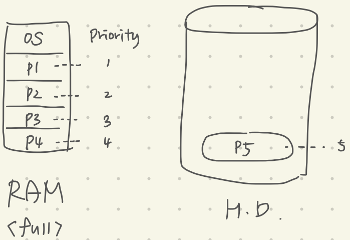
every process has a process control block & attributes

= **Context** of a process



H.D 내에는 new state인 여러개의 process가 존재한다.

1. which & how many processors in hard disk should be moved into the RAM? by **Long term scheduler**
2. which processor in RAM should access the CPU first? by **Short term scheduler** (= scheduler
3. which & how many processors in RAM should be moved back into the H.D? by **Medium term scheduler**



: P5가 priority가 높아 RAM에 들어가면 먼저 실행되겠지만 RAM에 남은 공간이 없는 상황

⇒ 기존에 RAM에 있던 process 中

swapping

low priority를 가진 process를 H.D에 옮기고 → swap - out

H.D에 있는 higher priority process를 RAM에 옮김 → swap - in

· **Context switching**

CPU에서 실행되던 process보다 높은 priority의 process가 RAM에 있을 때

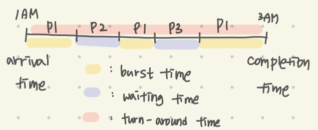
CPU에서 처리되던 process의 context는 남긴 채 higher priority process를 실행하는 것

(그 뒤 해당 작업이 끝나면 이전 process를 실행하던 지점 (context 정보를 가지고)부터 다시 실행하도록)

Various times of a process

1. **Arrival time** : time when process has arrived RAM
2. **Burst time** (= execution time) :
3. **Completion time** : time when process has completed execution & can be removed from RAM
4. **Turn-around time** : time between arrival time to completion time.
5. **Waiting time** : waiting time for execution or I/O (process is idle at this moment)
   ① completion time - arrival time
   ② burst time + waiting time + I/O time
6. **Response time** : 나중에 다룰 것
7. **I/O time** : amount of time spent in reading & writing I/O

ex) Various times for process 1



(cf)

- **point in time**
  arrival time, completion time
- **duration in time**
  burst time, turn-around time, waiting time, response time

Types of scheduling algorithms

CPU scheduling algorithms

- **Preemtive scheduling algorithms** : preempt the process
- **Non-preemtive scheduling algorithms** : CPU에서 process를 execute 하는 중일 때, RAM에 higher priority process가 생겨도 execution을 멈추지 않는 방식

· CPU scheduling algorithms are applied only to processes which are in ready state.
Process which are in I/O state will be "blocked" and will not be considered by scheduling algorithms
I/O state = block state
: P1 - running state / P2, P3 - ready state / P4 - I/O state 일 경우
P4의 priority가 P2, P3보다 높더라도 P1 실행이 끝난 뒤 P2, P3 중에 priority가 높은 process가 실행된다.

SJF scheduling algorithm

Shortest Job First scheduling algorithm

: Among the arrived process, process with the least burst time will be given preference (higher priority)

non - preemptive scheduling & priority based algorithm

ex1)

process Id	1	2	3	4	5
arrival time	2	3	4	6	8
burst time	3	2	3	1	2

Q) Find the average TAT, WT, Throughput and schedule length.

S)

Idle	P1	P2	P4	P5	P3

0 2 5 7 8 10 13

이 시간이전에
arrive 한 process들 中
burst time이 짧은 것이 다음에 실행됨

A)

CT	5	7	13	8	10	
· TAT (CT-AT)	3	4	9	2	2	→ avg = 4
· WT (TAT-BT)	0	2	6	1	0	

└ in this case we don't assume I/O time

· Schedule length = CT of last process - AT of first process = 11

· Throughput : number of processed executed in unit time = Number of processes / schedule length

= 5/11 = Processes / unit time

ex2)

process Id	1	2	3	4	5
arrival time	1	2	3	4	5
burst time	7	5	1	2	8

Q) Find the average TAT, WT, Throughput and schedule length.

S)

	P1	P3	P4	P2	P5

0 1 8 9 11 16 24

CT	8	16	9	11	24	
· TAT (CT-AT)	7	14	6	7	19	→ avg = 10.6
· WT (TAT-BT)	0	9	5	5	11	

· scheduling length = 23

· throughput = 5/23

SRTF Scheduling algorithm

Shortest Remaining time first scheduling algorithm : preemptive version of SJF

: 이미 process가 실행 중이더라도 새로운 (RAM에 있는) process의 실행에 필요한 총 시간이 현재 process 에서 남은 시간보다 짧다면, 먼저 실행
단, 모든 process가 ready queue에 있다면 SJF와 같은 방식으로 실행

ex1)

process Id	1	2	3	4	5	6
arrival time	0	1	2	3	4	5
burst time	~~7~~ ~~6~~ 0	~~5~~ ~~4~~ 0	~~3~~ ~~2~~ ~~1~~ 0	~~1~~ 0	~~2~~ 0	~~1~~ 0

Q) Find the average TAT, WT, Throughput and schedule length.
(같은 시간이 존재할 경우 먼저 arrive한것 먼저 실행할 것)

S)

P1	P2	P3	P4	P3	P3	P6	P5	P2	P1

0 1 2 3 4 5 6 7 9 13 19

A)

CT	19	13	6	4	9	7	← last completed process부터 계산하는 것이 좋다.
TAT (CT-AT)	19	12	4	1	5	2	→ avg = 43/6
WT (TAT-BT)	12	7	1	0	3	1	

└ in this case we don't assume I/O time

- Schedule length = CT of last process − AT of first process = 19
- Throughput : number of processed executed in unit time = Number of processes / schedule length
  = 6/19 = Processes / unit time

*** Response time** : waiting time of certain process until it gets to the CPU for the first time

process Id	1	2	3	4
arrival time	0	15	30	45
burst time	20	25	10	15

SJF

P1	P2	P3	P4

0 20 45 55 70

SRTF

P1	P1	P2	P3	P2	P2	P4

0 15 20 30 40 45 55 70

SJF RT	0	5	15	10
SRTF RT	0	5	0	10

⇨ In any non-preemptive scheduling algorithm, Response time = waiting time, but it may not be true in preemptive scheduling algorithm.

ex2)

Problem :

Consider the following CPU processes with arrival times (in milliseconds) and length of CPU bursts (in milliseconds) as given below:

Process	Arrival Time	Burst Time	CT	TAT	WT	RT
P1	0	~~7~~ ~~4~~ 0	12	12	5	0
P2	3	~~3~~ ~~1~~ 0	6	3	0	0
P3	5	~~5~~ 0	17	12	7	7
P4	6	~~2~~ 0	8	2	0	0

P1	P2	P2	P4	P1	P3

0 3 5 6 8 12 17

If the pre-emptive shortest remaining time first scheduling algorithm is used to schedule the processes, then the average waiting time across all processes is __3__ milliseconds.

An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process	Arrival Time	Burst Time	CT	TAT	WT	RT
P1	0	~~12~~ ~~10~~ 0	27	27	15	0
P2	2	~~4~~ ~~3~~ 0	6	4	0	0
P3	3	~~6~~ ~~4~~ 0	12	9	3	3
P4	8	~~5~~ 0	17	9	4	4

P1	P2	P2	P3	P3	P4	P1

0 2 3 6 8 12 17 27

The average waiting time (in milliseconds) of the processes is __5.5__.

FCFS scheduling algorithm

First Come First Served Scheduling Algorithm

The process which has the least arrival time will be scheduled first

non-preemptive # not priority based (not depending on a property of a process ex. burst time)

priority based algorithm의 단점은 나중에 알아볼것

process Id	1	2	3	4	5
arrival time	0	1	2	3	4
burst time	4	3	1	2	5

P1	P2	P3	P4	P5
0–4	4–7	7–8	8–10	10–15

CT	4	7	8	10	15
TAT	4	6	6	7	11
WT	0	3	5	5	6
RT	0	3	5	5	6

} in non-preemptive scheduling, WT=RT

Context switching : preempt the process and schedule some other process

사실은 context switching에 시간이 소요됨 ⇒ decreases the efficiency of CPU = slower
(degrade the performance)

$$\left[\ \text{CPU efficiency} = \frac{\text{Useful time}}{\text{Total time}}\ \right]$$

ex) context switching overhead = 1

process Id	1	2	3	4	5	6
arrival time	0	1	2	3	4	5
burst time	3	2	1	4	5	2

Context switching : CPU is idle

	P1	C	P2	C	P3	C	P4	C	P5	C	P6
0–1	1–4	4–5	5–7	7–8	8–9	9–10	10–14	14–15	15–20	20–21	21–23

CPU efficiency = $17/23$

CPU inefficiency = $6/23$

CT	4	7	9	14	20	23
TAT	4	6	7	11	16	18
WT	1	4	6	7	11	16
RT	1	4	6	7	11	16