운영체제 2차 과제 – CPU Scheduling 문제 풀이

2013130890
영어영문학과
김승태

5.1 A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given *n* processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of *n*.

Answer : n! (n 팩토리얼 = n × n – 1 × n – 2 × … × 2 × 1)

5.2 Explain the difference between preemptive and nonpreemptive scheduling.

Answer :

Preemptive : 선점하다라는 뜻으로 Running -> Waiting / Waiting -> running으로 가던 현재 프로세스에 변화를 주는 것. 비교 요소에서 우선순위가 있으면 현재 작업 중인 프로세스를 새로운 프로세스로 대체하는 것입니다.

Nonpreemptive : 현재 작업중인 프로세스의 실행이 완료될 때까지 (중간에 interrupted 없이) 프로세스의 실행을 보장하는 것입니다.

5.3 Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use **nonpreemptive scheduling**, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

Burst Time과 같이 주어진 것이 Arrival Time or Priority 등이 따라올 수 있다.

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm? **Answer : [8 + (12 – 0.4) + (13-1)] / 3 = 10.53 → arrival time이 함께 고려되야함**
b. What is the average turnaround time for these processes with the SJF scheduling algorithm? **Answer : (8 + (9 - 1) + (13 – 0.4)) / 3 = 9.53**
c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process *P*1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes *P*1 and *P*2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as ***future- knowledge scheduling***.

Answer : [(2-1) + (6-0.4) + 14] / 3 = 6.86

5.4 Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

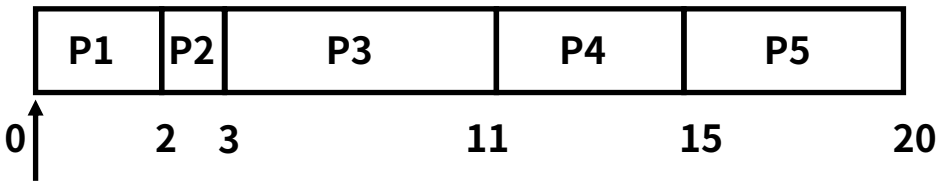
b. What is the turnaround time of each process for each of the scheduling algorithms in part a.?

c. What is the waiting time of each process for each of these scheduling algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

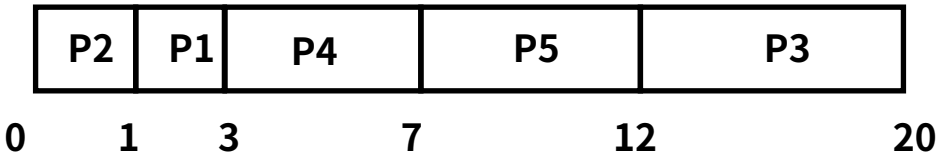
A번 문제 정답(Gantt Chart)

FCFS

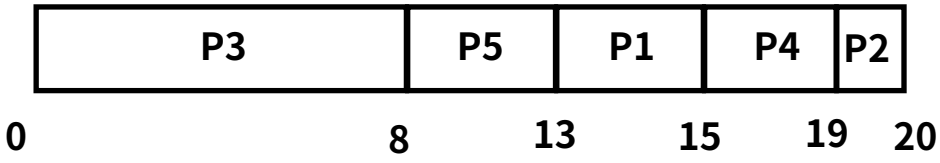


P1 ~ P5 모두 0에 도착

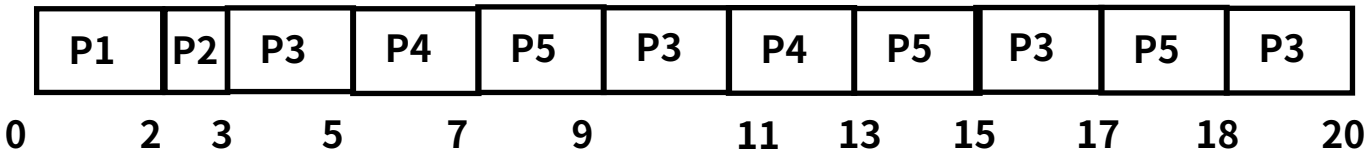
SJF



Nonpreemptive Priority



Round Robin (Quantum=2)



B번 문제 정답 (turnaround time of each process)

	FCFS	SJF	Nonpreemptive Priority	Round Robin
P1	2	3	15	2
P2	3	1	20	3
P3	11	20	8	20
P4	15	7	19	13
P5	20	12	13	18

C번 문제 정답 (waiting time of each process)

	FCFS	SJF	Nonpreemptive Priority	Round Robin
P1	0	1	13	0
P2	2	0	19	2
P3	3	12	0	12
P4	11	3	15	9
P5	15	7	8	13

D번 문제 정답 (minimum waiting time)

	FCFS	SJF	Nonpreemptive Priority	Round Robin
average	6.2	4.6	11	7.2

Answer : SJF is the shortest waiting time

5.5 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm.

Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an **idle task** (which consumes no CPU resources and is identified as P_{idle}). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

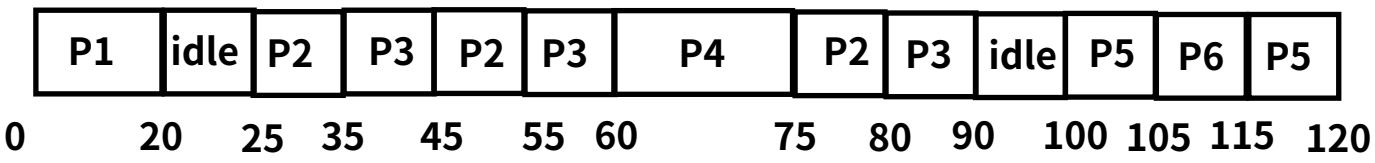
a. Show the scheduling order of the processes using a Gantt chart.

b. What is the turnaround time for each process?

c. What is the waiting time for each process?

d. What is the CPU utilization rate?

A. Gantt Chart



B & C Answer

	Turnaround Time	Waiting Time
p1	20 – 0 = 20	0
p2	80 - 25 = 55	10 + 20 = 30
p3	90 – 30 = 60	5 + 10 + 20 = 35
p4	75 – 60 = 15	0
p5	120 – 100 = 20	115 – 105 = 10
p6	115 – 105 = 10	0

D Answer : 105 / 120 = 87.5 %

5.6 What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

Answer :주어진 프로세스 burst time과 비슷하거나 크게 time-quantum을 설정하면 잦은 context switch 없이 작업을 진행할 수 있으며, (굳이 CPU Burst가 큰 작업을 context switching을 자주 하면서 얻게 되는 이득이 없다) 적절한 수준의 time-quantum을 지정하면 프로세스 수행에 대한 효율성을 높일 수 있다

5.7 Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on.

These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?

a. Priority and SJF

b. Multilevel feedback queues and FCFS

c. Priority and FCFS

d. RR and SJF

Answer :

a. SJF에선 CPU Burst Time이 작은 값이 Priority가 높다

b. 새로운 job이 들어올 땐, FCFS가 적용될 수 있으며 이후 해당 프로세스에 업무가 몰렸을 경우, 다른 프로세스로 로드 밸런싱을 진행할 때, 다시 FCFS가 적용되는 형태로 Multilevel feedback queue에 사용될 수 있습니다.

c. FCFS에선 먼저 들어온 작업에 대해서 높은 Priority를 제공합니다.

d. 특별한 연관성을 찾아보기 힘드나, 양쪽 모두 preemptive 를 적용할 수 있는 점이 공통점으로 볼 수 있겠습니다

5.8 Suppose that a CPU scheduling algorithm favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

Answer :

I/O Bound Program을 실행하면 CPU burst 크기는 작지만 상대적으로 많은 횟수가 실행되는 형태로 수행되기 때문에 I/O bound program이 유리해 보이지만, I/O Bound Program이 I/O를 수행하기 위해 CPU를 상대적으로 자주 포기하기 때문에 CPU Bound Program이 중단되지는 않습니다.

5.9 Distinguish between PCS and SCS scheduling.

Answer :

PCS는 Process Contention Scope의 의미로 동일한 프로세스에 속한 스레드 사이에 CPU 경쟁을 하게 되는데 프로세스 경쟁 범위로 이해할 수 있습니다.

SCS는 System Contention Scope의 의미로 CPU 상의 어느 커널에 스레드를 할당할 것인지 결정하는 업무를 수행할 때 사용합니다.

5.10 The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the number, the lower the priority. The scheduler recalculates process priorities once per second using the following function:

$$\text{Priority}=(\text{recent CPU usage}/2)+\text{basePriority}=(\text{recent CPU usage}/2)+\text{base}$$

where base = 60 and *recent CPU usage* refers to a value indicating how often a process has used the CPU since priorities were last recalculated.
Assume that recent CPU usage for process P_1 is 40, for process P_2 is 18, and for process P_3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

Aanswer :

위 방정식의 계산에 따라 현재 base값은 60이므로, 프로세스들에 대한 우선순위는 아래와 같이 계산됩니다.

P1 priority = (40/2) + 60 → 80
P2 priority = (18/2) + 60 → 69
P2 priority = (10/2) + 60 → 65

그러나 연산 결과의 우선순위가 변경되는 사항은 없기 때문에 traditional UNIX 스케줄러는 CPU bound process에 상대적으로 우선순위를 낮게 줄 것으로 예측됩니다.

5.11 Of these two types of programs:

a. I/O-bound

b. CPU-bound

which is more likely to have voluntary context switches, and which is more likely to have nonvoluntary context switches? Explain your answer.

Answer

인터럽트가 발생하는 상황 자체가 비자발적

I/O Bound : 프로세스가 시스템 호출을 하고 해당 커널 함수 안에서 입출력 요구 등을 하여 프로세스가 대기 상태로 전환하며 자발적으로 문맥 교환(voluntary context switch)를 하는 경우

CPU Bound : 인터럽트의 처리 및 시스템 호출 완료 직후 사용자 모드로의 복귀 이전에 스케줄러가 수행되며 이 때 우선순위가 현재 프로세스 보다 높은 프로세스가 있으면 비자발적 문맥 교환(involuntary context switch)이 일어난다.

5.12 Discuss how the following pairs of scheduling criteria conflict in certain settings.

a. CPU utilization and response time

b. Average turnaround time and maximum waiting time

c. I/O device utilization and CPU utilization

Answer

a. CPU 사용률 및 응답 시간: context switch와 관련된 오버헤드가 최소화되는 경우 CPU 사용률이 증가합니다. context switch를 자주 수행하지 않음으로써 컨텍스트 전환 오버헤드를 줄일 수 있지만 그러나 이로 인해 프로세스에 대한 응답 시간이 늘어날 수 있습니다.

b. Average turnaround time and maximum waiting time: 평균 소요 시간은 가장 짧은 작업을 먼저 실행하여 최소화할 수 있습니다. 그러나 이러한 스케줄링 정책은 장시간 실행되는 작업을 중단시켜 대기 시간을 늘릴 수 있습니다.

c. I/O 디바이스 사용률 및 CPU 사용률: context switch를 수행하지 않고 CPU bound 작업을 실행하여 CPU 사용률을 최대화합니다. I/O 장치 활용도는 I/O bound 작업을 실행할 준비가 되는 즉시 스케줄링하여 컨텍스트 스위치의 오버헤드를 발생시킴으로써 극대화가 될 것입니다.

5.13 One technique for implementing **lottery scheduling** works by assigning processes lottery tickets, which are used for allocating CPU time. Whenever a scheduling decision has to be made, a lottery ticket is chosen at random, and the process holding that ticket gets the CPU. The BTV operating system implements lottery scheduling by holding a lottery 50 times each second, with each lottery winner getting 20 milliseconds of CPU time (20 milliseconds × 50 = 1 second). Describe how the BTV scheduler can ensure that higher-priority threads receive more attention from the CPU than lower-priority threads.

Answer:

Lottery Ticket에 더 높은 우선순위 프로세스를 할당함으로써 높은 우선순위 스레드를 받는 것으로 추정됩니다.

5.14 Most scheduling algorithms maintain a **run queue**, which lists processes eligible to run on a processor. On multicore systems, there are two general options: (1) each processing core has its own run queue, or (2) a single run queue is shared by all processing cores. What are the advantages and disadvantages of each of these approaches?

Advantage

(1) : 각 프로세스 코어마다 고유의 run-queue를 가질 때의 장점은 스케줄러가 2개 이상의 프로세서에서 동시 실행 중일 때 단일 실행 대기열에서 경쟁이 없다는 점입니다. 프로세스 코어에서 스케줄링을 진행할 때, 스케줄러는 해당 코어의 전용 run-queue만 보는 장점을 얻습니다.

(2) Asymmetric Multiprocessing으로 마스터 cpu는 커널 코드 실행하고, 나머지는 유저 코드를 실행하는 형태기 때문에 비교적 구현이 쉽습니다.

Disadvantage

(1) : 실행 중인 업무가 끝나면, 새로운 업무가 run-queue에 없을 경우 프로세서가 아무것도 하지 않는다는 점

(2) : 여러 코어가 하나의 run-queue를 사용하기 때문에 코어 사이에 경쟁이 있을 수 있어 비효율적일 수 있습니다.

5.15 Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?

a. $\alpha = 0$ and $\tau_0 = 100$ milliseconds

b. $\alpha = 0.99$ and $\tau_0 = 10$ milliseconds

Answer :

a. 가장 최근에 수행되었던 프로세스의 cpu 버스트 시간은 고려하지 않고, 가장 처음에 수행되었던 프로세스의 cpu 버스트 시간은 100 밀리초였다.

b. 가장 최근에 수행되었던 프로세스의 cpu 버스트 시간을 중점적으로 고려하고, 가장 처음에 수행되었던 프로세스의 cpu 버스트 시간은 10 밀리초였다.

5.16 A variation of the round-robin scheduler is the regressive round-robin scheduler. This scheduler assigns each process a time quantum and a priority. The initial value of a time quantum is 50 milliseconds. However, every time a process has been allocated the CPU and uses its entire time quantum (does not block for I/O), 10 milliseconds is added to its time quantum, and its priority level is boosted. (The time quantum for a process can be increased to a maximum of 100 milliseconds.) When a process blocks before using its entire time quantum, its time quantum is reduced by 5 milliseconds, but its priority remains the same. What type of process (CPU-bound or I/O-bound) does the regressive round-robin scheduler favor? Explain.

Answer:

이 스케줄러는 CPU bound된 프로세스가 전체 time quantum을 소비할 때마다 우선 순위 상승뿐만 아니라 더 긴 time quantum으로 보상을 받기 때문에 선호할 것이다. 이 스케줄러는 전체 time quantum을 사용하기 전에 I/O를 차단하기 쉽기 때문에 I/O bound 프로세스에 불이익을 주지는 않지만 우선 순위는 그대로 유지됩니다.

5.17 Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Priority
P_1	5	4
P_2	3	1
P_3	1	2
P_4	7	2
P_5	4	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5, all at time 0.
a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

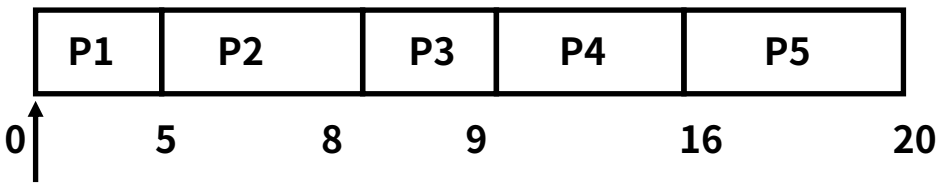
c. What is the waiting time of each process for each of these scheduling algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

다음 장에서 계속

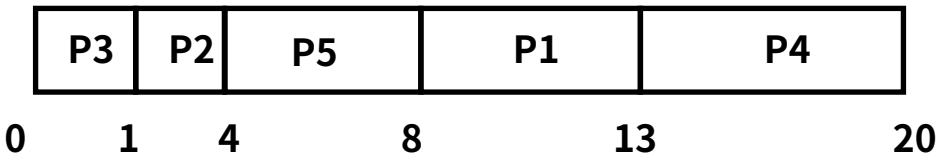
A번 문제 정답(Gantt Chart)

FCFS

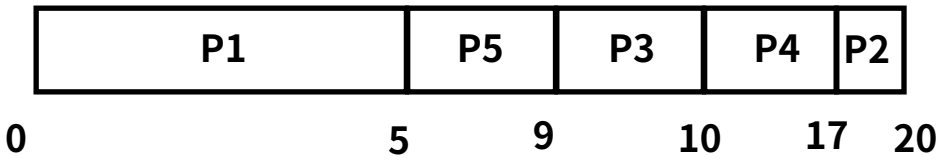


P1 ~ P5 모두 0에 도착

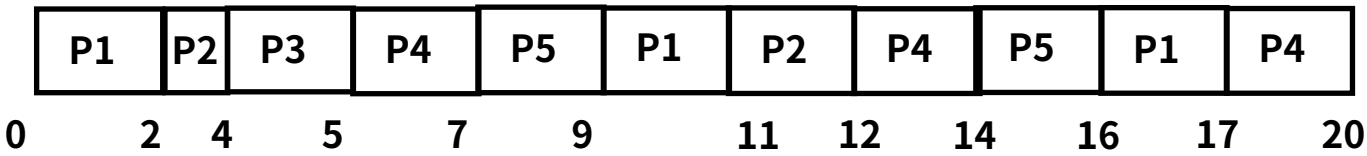
SJF



Nonpreemptive Priority



Round Robin (Quantum=2)



B & C Answer

FCFS

	Turnaround Time	Waiting Time
p1	5	0
p2	8	5
p3	9	8
p4	16	9
p5	20	16

Average Turnaround Time : 11.6
Average Waiting Time : 7.6

SJF

	Turnaround Time	Waiting Time
p1	13	8
p2	4	1
p3	1	0
p4	20	13
p5	8	4

Average Turnaround Time : 9.2
Average Waiting Time : 5.2 (the shortest waiting time)

Non-preemptive Priority

	Turnaround Time	Waiting Time
p1	5	0
p2	20	17
p3	10	9
p4	17	10
p5	9	5

Average Turnaround Time : 12.2
Average Waiting Time : 8.2

RR (quantum=2)

	Turnaround Time	Waiting Time
p1	17	7 + 5 = 12
p2	12	2 + 7 = 9
p3	5	4
p4	20	5 + 5 + 3 = 13
p5	16	7 + 5 = 12

Average Turnaround Time : 14
Average Waiting Time : 10

D번 문제 정답 (minimum waiting time)

	FCFS	SJF	Nonpreemptive Priority	Round Robin
average	7.6	5.2	8.2	10

Answer : SJF is the shortest waiting time

5.18 The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm.

Process	Burst Time	Priority	Arrival
P_1	8	15	0
P_2	3	20	0
P_3	4	20	20
P_4	4	20	25
P_5	5	5	45
P_6	5	15	55

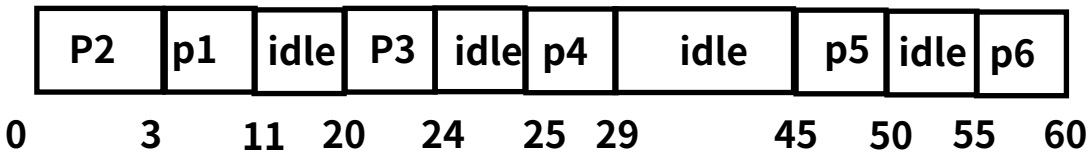
Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. The scheduler will execute the highest-priority process. For processes with the same priority, a round-robin scheduler will be used with a time quantum of 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a. Show the scheduling order of the processes using a Gantt chart.

b. What is the turnaround time for each process?

c. What is the waiting time for each process?

A. Gantt Chart



B & C Answer

	Turnaround Time	Waiting Time
p1	11-0 =11	3
p2	3-0 = 3	0
p3	24-20 = 4	0
p4	29-25 = 4	0
p5	50 – 45 = 5	0
p6	60-55 = 5	0

5.19 The `nice` command is used to set the nice value of a process on Linux, as well as on other UNIX systems. Explain why some systems may allow any user to assign a process a nice value >= 0 yet allow only the root (or administrator) user to assign nice values < 0.

Answer:

Nice value값이 0보다 작은 경우에 상대적으로 더 높은 우선순위가 할당되며, 이러한 시스템은 루트 외의 프로세스에 높은 우선순위가 할당되는 것을 대개 허용하지 않습니다

5.20 Which of the following scheduling algorithms could result in starvation?

a. First-come, first-served

b. Shortest job first

c. Round robin

d. Priority

Answer: b. Shortest job first / d. priority

b. SJF 및 우선순위 기반 스케줄링 알고리즘에서 많은 소규모 프로세스가 실행되고있는 busy 시스템에서 starvation이 발생할 수 있습니다.

우선순위가 낮은 작업이 뒤로 밀리면서 작업량이 작은 값이 계속 들어올 경우, 작업량이 큰 값은 영원히 실행이 안될 수 있습니다.

5.21 Consider a variant of the RR scheduling algorithm in which the entries in the ready queue are pointers to the PCBs.

a. What would be the effect of putting two pointers to the same process in the ready queue?

b. What would be two major advantages and two disadvantages of this scheme?

c. How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

Answer

(a) 프로세스가 2배 더 많이 실행되어 우선순위가 높아지는 효과를 얻을 수 있습니다.

(b)
- Advantages
  1. 유저에게 더 중요한 프로세스에 대해 우선순위를 제공할 수 있게 도와줍니다.
  2. 우선순위가 낮은 프로세스에 대해 starvation이 발생하는 것을 방지합니다.
- Disadvantage
  1. Context switching가 더 큰 효과를 발휘하게 될 것입니다.
  2. 상대적으로 shorter한 작업들은 뒤로 밀려 피해를 볼 수 있습니다.

(c) Quantum time을 각 프로세스 별로 다르게 적용하는 것을 허용하는 것과 같은 효과를 가져옵니다.

5.22 Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond

b. The time quantum is 10 milliseconds

Answer

a. 스케줄링된 프로세스에 관계없이 스케줄러는 모든 컨텍스트 스위치에 대해 0.1밀리초의 컨텍스트 전환 비용을 발생시킵니다. 따라서 CPU 활용률이 1/1.1 * 100 = 91%가 됩니다.

b. I/O bound 작업은 Time Quantum의 1밀리초만 사용한 후 컨텍스트 스위치를 발생시킵니다. 따라서 모든 프로세스를 순환하는 데 필요한 시간은 10*1.1 + 10.1입니다(각 I/O 바인딩 작업이 1밀리초 동안 실행되었다가 컨텍스트 스위치 작업이 발생하지만 CPU 바인딩 작업은 컨텍스트 스위치를 발생시키기 전에 10밀리초 동안 실행됩니다). 따라서 CPU 사용률은 20/21.1 * 100 = 94%입니다.

5.23 Consider a system implementing multilevel queue scheduling. What strategy can a computer user employ to maximize the amount of CPU time allocated to the user's process?

Answer:

프로그램은 Time Quantums을 완전히 활용하지 않음으로써 할당되는 CPU 시간을 최대화할 수 있습니다. 할당된 Quantums의 많은 부분을 사용할 수 있지만, Quantum이 끝나기 전에 CPU를 포기하여(relinquish) 프로세스와 관련된 우선 순위를 증가시킬 수 있습니다.

5.24 Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α. When it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

a. What is the algorithm that results from $\beta > \alpha > 0$?
b. What is the algorithm that results from $\alpha < \beta < 0$?

Answer

a. FCFS(현재 작업 중인 프로세스의 우선순위가 높고, 이 프로세스 종료 이후 다음 프로세스가 실행)
b. LIFO(위의 순서와 정반대)

5.25 Explain the how the following scheduling algorithms discriminate either in favor of or against short processes:

a. FCFS

b. RR

c. Multilevel feedback queues

Answer

1.FCFS—긴 job 이후에 도착하는 짧은 작업의 대기 시간이 길어지기 때문에 짧은 작업을 차별합니다.
2.RR—모든 작업을 동등하게 처리(CPU 시간의 동일한 버스트 제공)하므로 짧은 작업은 먼저 완료되기 때문에 시스템에서 더 빨리 떠날 수 있다.
3.Multilevel feedback queues - RR 알고리듬과 유사하게 작동하며, 짧은 작업에 대해 호의적으로 구별한다.

5.26 Provide a specific circumstance that illustrates where rate-monotonic scheduling is inferior to earliest-deadline-first scheduling in meeting process deadlines?

Answer

여기 2개의 프로세스가 있다고 가정해보겠습니다.

(p1 = 50, t1 = 20 / p2 = 100, t2 = 35)

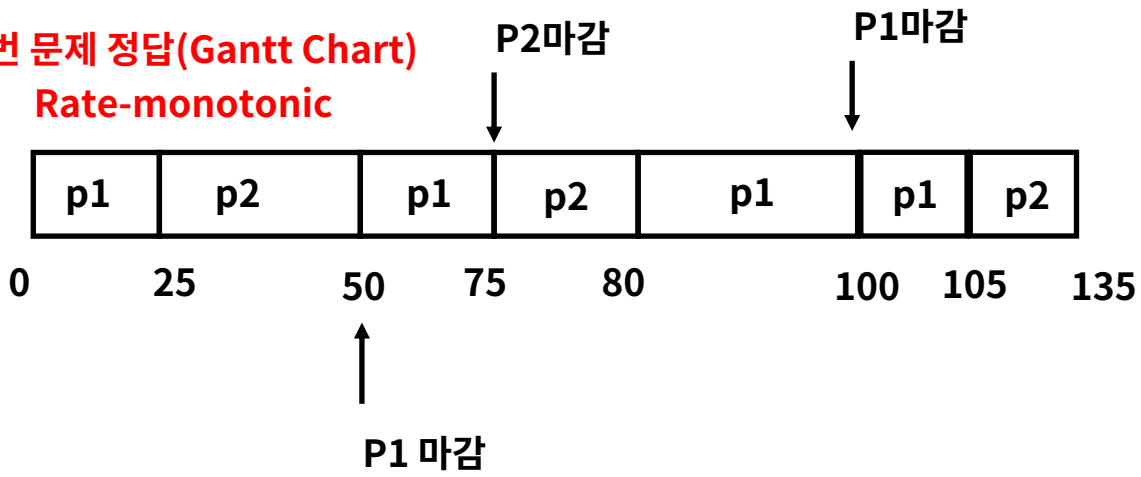
P1은 p2보다 높은 우선순위를 부여받았고, 스케줄링 작업은 rate-monotonic 스케줄링을 사용하게 될 때, p1은 t=0에 스케줄을 받을 것이고, p2는 t=20일 때 작업을 할당 받은 후, p1이 다시 작업을 할당 받는 t=50까지 작업을 진행하게 될 것입니다. 그리고 p2는 다시 t=70부터 작업이 수행될 것입니다. 이 경우 p2의 작업이 마감에 맞게 진행되지 않는 문제가 발생합니다.

그러나 EDF를 사용하면 데드라인이 빠른 작업이 우선권이 있고, 작업이 중간에 끊기지 않기 때문에 t=0에 p1시작, t=20에 p2시작, t=55에 p1 시작 등으로 이어지게 될 수 있습니다. 이 경우 데드라인을 맞추기가 수월하기 때문에 Rate-monotonic 스케줄링보다 효율적이라 할 수 있습니다.

5.27 Consider two processes, P_1 and P_2, where $p_1 = 50$, $t_1 = 25$, $p_2 = 75$, and $t_2 = 30$.
a. Can these two processes be scheduled using rate-monotonic scheduling? Illustrate your answer using a Gantt chart such as the ones in Figure 5.21–Figure 5.24.

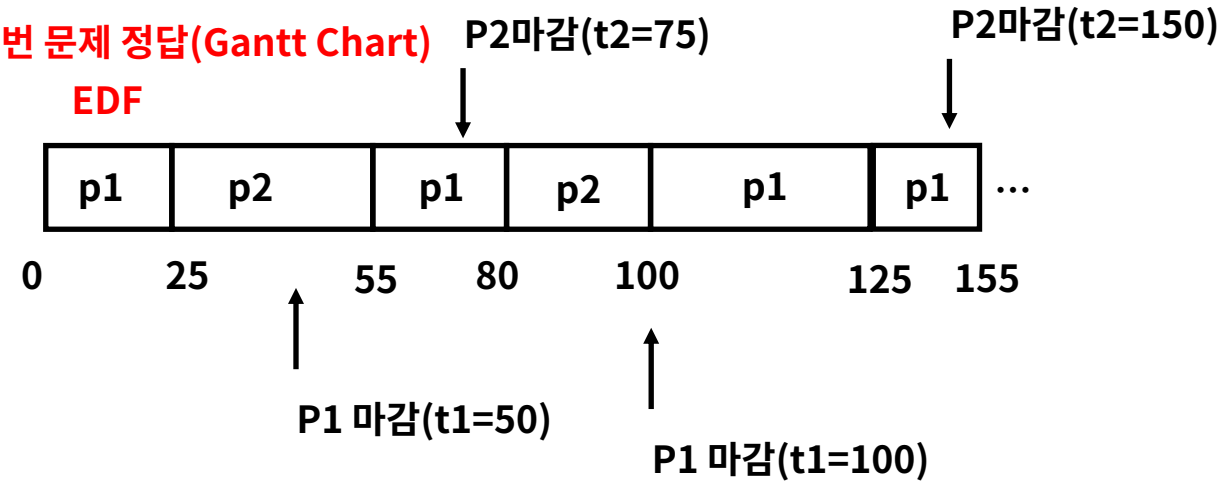
b. Illustrate the scheduling of these two processes using earliest-deadline-first (EDF) scheduling.

A번 문제 정답(Gantt Chart)
Rate-monotonic



P1의 값이 더 낮아 우선순위가 더 높게 부여되었다.
그러나 그것과 별개로 마감기한에 따라 preemptive가 빈번히 발생하면서 p2 프로세스가 적절히 수행되지 못하는 문제가 발생하는 것을 확인할 수 있다.

B번 문제 정답(Gantt Chart)
EDF



EDF를 적용할 경우 p1의 데드라인에 p2가 걸려있을 때도 실행을 보장하고, 반대의 경우도 보장하는 것을 볼 수 있는데, EDF를 사용하면 현 시점에서 가장 빠른 데드라인의 작업이 먼저 실행되기 때문에 Rate-monotonic에서 프로세스가 적절히 수행 안되던 것을 관리할 수 있다.