

Operating System Project-1

[MFQ Scheduling Simulation]

2014312455 도경희

목차

1. 프로젝트 요약

- 1) 목표
- 2) 고려사항 및 가정

2. 전략 및 소스코드 분석

- 1) 전략
- 2) MFQ.h

3. 결과 분석

- 1) 테스트파일 1
- 2) 테스트파일 2
- 3) 테스트파일 3
- 4) 테스트파일 4
- 5) 테스트파일 5

1. 프로젝트 요약

1) 목표

이번 과제의 목표는 MFQ 스케줄링 기법을 시뮬레이션 하는 것이다.

2) 고려사항 및 가정

(1) 고려사항:

본 과제에서 MFQ는 4개의 Ready Queue를 가진다. Q0은 Time quantum(TQ) 2인 Round Robin(RR), Q1은 TQ 6인 RR, Q2는 Shortest Remaining Time Next, Q3는 First Come First Served(FCFS) 스케줄링 기법을 사용한다. 각 프로세스가 최초로 진입할 queue는 인풋파일에 명시되어 있으며, Qi에서 preemption된 프로세스는 Qi+1로 진입하고, Qi에서 실행한 프로세스가 IO burst를 마치고 wakeup되는 경우는 Qi-1로 진입한다. 우선순위는 $Q0 > Q1 > Q2 > Q3$ 의 순이고 스케줄링 순서 또한 이를 따른다. 또, TQ를 할당받아 실행중인 프로세스는 TQ가 끝날 때까지는 preemption되지 않는다.

본 과제에 사용한 코드는 gcc compatible하지만, 코드의 가독성을 위해 MS visual studio2017를 사용한다. gcc환경에서의 실행결과는 본 보고서의 결과분석 파트에 별도로 첨부한다.

(2) 가정:

- ㄱ) 프로세스가 RQ에 있는 상태로 cpu의 burst time이 지나갈 경우 WT이 증가한다.
- ㄴ) SRTN에서는 Burst time이 모두 알려져 있다고 가정, burst time estimation은 하지 않는다.
- ㄷ) Q0, Q1에서 스케줄링받은 프로세스는 TQ를 다 채우기 전까지는 preemption되지 않는다.
- ㄹ) Q2에서 할당받은 프로세스는 Q2에 할당된 프로세스에 의해서만 preemption된다.
- ㅁ) Q3은 non-preemptive하다.
- ㅂ) 입력으로는 프로세스의 정보를 AT순으로 준다.
- ㅅ) 프로세스의 모든 burst cycle을 마친 프로세스는 결과 출력을 위해 잠시 zombieList라는 저장공간에 대기하는데, 이때 순서는 프로세스가 종료되는 순서대로이다.
- ㅇ) 여러 개의 Process가 동시에 하나의 RR과 FCFS RQ로 진입할 때, 프로세스의 우선순위는 모두 같으나, 본 프로젝트는 인풋파일에 입력된 순서(RQ로 초기 진입하는 경우), asleep상태로 먼저 전환된 순서(IO burst를 완료하고 RQ로 진입하는 경우)대로 처리한다.

2. 전략 및 소스코드 분석

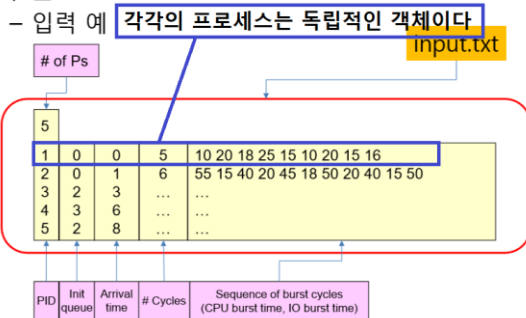
1) 전략

(1) 문제를 작은 부분들로 나누기(구조체)

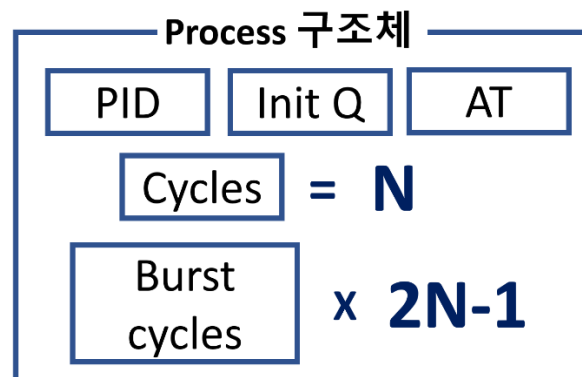
ㄱ) 프로세스

우선, 과제에서는 프로세스의 형태가 지정되어 있다. 그림(그림2-1)에 명시되어 있듯이 각각의 프로세스는 PID, Init queue(Init Q), Arrival time(AT), Cycles, sequence of burst cycles를 가진다. 또, 각각의 프로세스는 공유하는 자원이 없으므로, 구조체의 형태로 각각의 프로세스를 분리할 수 있다(그림2-2). 구조체의 형태에서 참고할 점은 cycles의 수를 n 이라 할 때, sequence of burst cycles는 $2n-1$ 의 숫자가 연속으로 나온다는 점이다.

• 구현



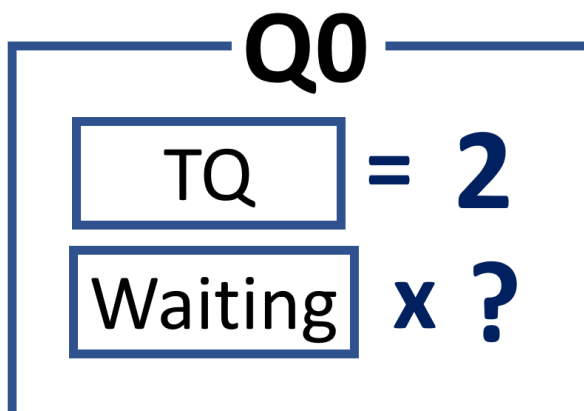
<그림 2-1. OS-2019-1-Project1 안내서>



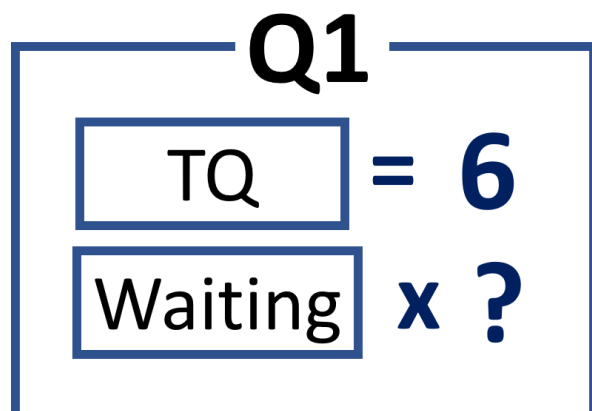
<그림 2-2. 프로세스 구조체 뼈대>

ㄴ) Q0 & Q1

Q0는 TQ가 2인 RR스케줄링 기법을 사용한다. 즉, preemption조건인 TQ를 저장할 공간과, Ready상태에 있는 프로세스를 담을 공간이 필요하다(그림2-3). Q1은 TQ가 6이라는 것 이외에는 Q0과 동일하다(그림 2-4).



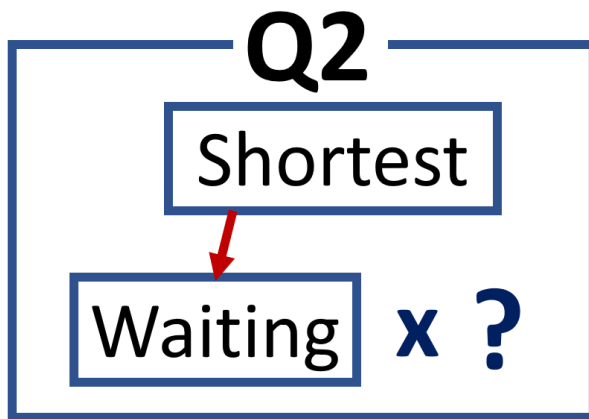
<그림 2-3. Q0 구조체 뼈대>



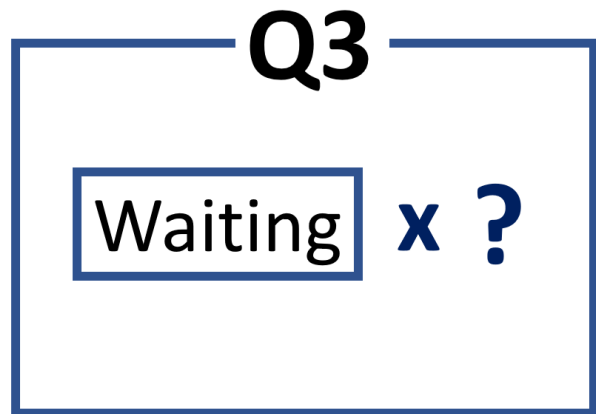
<그림 2-4. Q1 구조체 뼈대>

ㄷ) Q2

Q2는 SRTN기법을 사용하는 RQ이다. Q2는 preemption조건이 Q0, Q1과는 다르다. Q2에서 스케줄링된 프로세스보다 더 적은 burst time을 가지는 프로세스가 Q2에 존재할 때 preemption한다. 따라서, 가장 적은 burst time을 가지는 프로세스를 가리킬 공간과 대기하고 있는 프로세스를 담을 공간이 필요하다(그림 2-5).



<그림 2-5. Q2 구조체 뼈대>



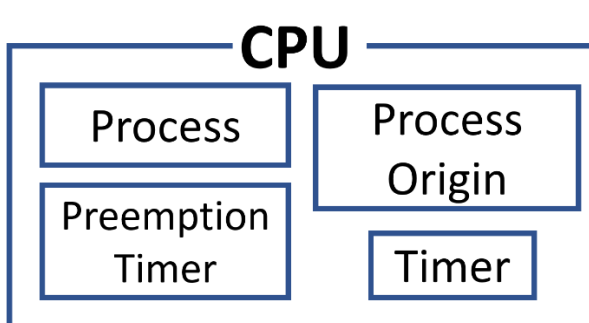
<그림 2-6. Q3 구조체 뼈대>

ㄹ) Q3

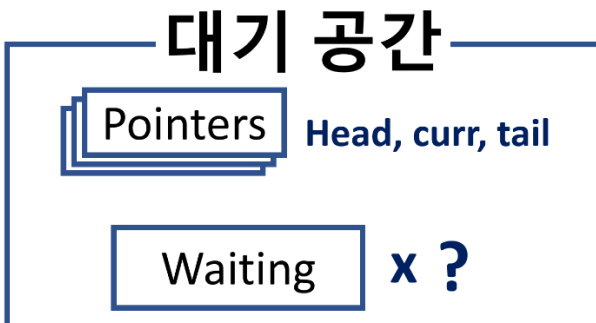
Q3는 FCFS기법을 사용하는 RQ이다. preemption조건이 없기 때문에 preemption조건을 저장할 공간은 필요하지 않다. 대기하는 프로세스를 담을 공간만 있으면 된다(그림 2-6).

ㄴ) CPU

CPU는 RQ에서 스케줄링 받은 프로세스들이 실행되는 공간이다. 따라서 CPU에는 프로세스를 저장할 공간이 있어야 하고, 프로세스의 preemption조건을 저장할 공간이 필요하다. 또, CPU는 preemption되었을 때에 프로세스를 다른 RQ로 보내야 하기 때문에 프로세스가 어느 RQ에서 왔는지에 대한 정보 또한 저장해야 한다(그림 2-7).



<그림 2-7. CPU 구조체 뼈대>



<그림 2-8. 대기 공간 구조체 뼈대>

ㄹ) RQ이외의 대기공간

RQ CPU이외의 공간이 두 가지 필요하다. Cpu burst이 끝난 프로세스가 I/O burst time이 끝날 때 까지 대기할 공간(asleepList), 그리고 모든 burst time을 소진한 프로세스가 결과 출력이 끝나기 전까지 잠깐 머무를 공간(zombieList)이 필요하다. 각각의 공간은 원소의 정보를 읽고, 때로는 수정해야 하기 때문에 원소에 접근이 가능해야 한다(그림 2-8).

ㄱ) 바구니

두 가지 형태의 바구니가 필요하다. 위 그림에서 보았듯 burst time을 묶어서 저장할 바구니, 그리고 즉 한 공간에 존재하는 프로세스들을 담을 바구니가 있어야 부품들을 체계적으로 관리할 수 있다. 이 때, 부품의 수는 가변적이므로 Linked-list를 사용한다. Burst time 바구니의 경우, burst time을 모두 소진하여야 다음 burst time에 접근할 수 있으므로 단방향, 프로세스 바구니의 경우 프로세스를 다른 곳으로 이동시켜야 하기 때문에 양방향으로 설계한다.

(2) 나누어진 부분들을 연결하기(함수)

ㄱ) 준비 단계

Program cycle이전에 해야 할 일을 정리하면 다음과 같다. 먼저, 프로세스의 정보를 취합해서 프로세스리스트를 만들어야 한다. 또 4개의 RQ를 만들고, CPU, asleepList, zombieList도 만들어야 한다. 이는 저장공간들을 만드는 함수, 그리고 만들어진 공간에 알맞은 값을 배분하는 함수를 만드는 것으로 실현 가능하다. 단, 저장공간의 크기를 모르기 때문에 동적으로 할당해야 한다.

ㄴ) 시간 소모 전 단계(Program cycle의 시작)

준비가 완료된 이후에는 어느 프로세스가 CPU를 할당받을지를 정해야 한다. $Q_0 > Q_1 > Q_2 > Q_3$ 의 우선순위에 따라 cpu를 할당받을 프로세스를 정하는 함수를 만들면 된다. 이때, Q2에서는 현재 cpu를 할당받은 프로세스가 Q2에서 스케줄링되었을 경우 burst time을 비교하여 이미 cpu를 할당받은 프로세스를 밀어낼 수 있다.

ㄷ) 시간 소모 단계(Program cycle의 중반부)

시간 소모 단계는 두 단계로 나뉜다. 먼저, asleepList에 있는 프로세스는 각각의 I/O Burst time을 소모해야 한다. 해당 프로세스 cycle의 I/O burst time을 모두 소모한 프로세스는 cpu burst time을 소모하기 위해서 해당 RQ로 이동하는데, 시간 소모 단계에서는 preemption이 일어나지 않는다. 다음으로 CPU를 할당받은 프로세스는 burst time을 소모해야 한다. 해당 프로세스 cycle의 Cpu burst time을 모두 소모한 프로세스는 다음 프로세스 cycle이 존재할 경우 asleepList로, 다음 프로세스 cycle이 존재하지 않을 경우 zombieList로 이동하게 해야 한다.

ㄹ) 시간 소모 이후 단계(Program cycle의 후반부)

Program cycle의 시간을 소모한 이후에는 Q0, Q1에서 스케줄링된 프로세스에 한하여 preemption여부를 검사한다. Q0과 Q1에서 스케줄링될 경우에는 TQ또한 같이 입력받기 때문에 TQ와 CPU의 timer를 비교하는 것을 통해 preemption여부를 결정할 수 있다.

ㄱ) 모든 프로세스가 burst time을 소모할 때까지 cycle 반복

모든 프로세스가 burst time을 모두 소진할 때까지 ㄴ)~ㄹ)의 과정을 반복한다. 반복문을 사용해야 하고, 끝나는 시점을 예측하기 어렵기 때문에 while/do while문을 사용해야 한다.

ㄴ) 결과 출력

burst time을 모두 소진한 프로세스는 zombieList로 이동한다. 모든 프로세스가 zombieList로 이동한 이후에는 zombieList를 처음부터 끝까지 살펴보면서 Turnaround time, Wait time을 출력하고, 이에 기반하여 Average Turnaround time, Average Wait time을 출력할 수 있다.

ㄷ) 정리 단계

준비 단계에서 동적으로 할당한 메모리를 해제해주어야 한다.

2) MFQ.h

본 보고서는 프로그램의 흐름을 알 수 있도록 헤더파일만을 설명한다. MFQSource.c와 MFQMain.c는 첨부파일을 통해 제공된다.

(1) 구조체

```
1  #ifndef MFQ_H
2  #define MFQ_H
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  //node to store burst time
7  typedef struct _BNode {
8      int burst;
9      struct _BNode* next;
10 } BNode;
11
12 //linked list to store nodes containing burst time
13 typedef struct _BList {
14     struct _BNode* head;
15     struct _BNode* curr;
16     struct _BNode* tail;
17 } BList;
18
```

```

19 //Process that 1)stores values from input
20 // 2) works as node in double linked list
21 // 3) stores values for result report
22 typedef struct _Process {
23     int PID; //Process ID
24     int Q; //Queue to enter
25     int AT; //Arival time
26     int TT; //Turnaround time
27     int WT; //Waiting time
28     int cycles; //if cycle is n, then there would be 2*n-1 sequences of bursts
29     struct _BList* bursts; //Linked list for bursts
30     struct _Process* next; //next Process
31     struct _Process* prev; //Previous Process
32 } Process;
33
34 //Process containers to be used as waiting queue, asleepList, zombieList
35 typedef struct _ProcessList {
36     struct _Process* head; //First element of list
37     struct _Process* curr; //Current element
38     struct _Process* tail; //Last element
39 } ProcessList;
40
41 //RQ with time quantum of 2
42 typedef struct _Q0 {
43     int TQ;
44     struct _ProcessList* waiting;
45 } Q0;
46
47 //RQ with time quantum of 6
48 typedef struct _Q1 {
49     int TQ;
50     struct _ProcessList* waiting;
51 } Q1;
52
53 //SRTN RQ
54 typedef struct _Q2 {
55     struct _Process* shortestNow;
56     struct _ProcessList* waiting;
57 } Q2;
58
59 //FCFS RQ
60 typedef struct _Q3 {
61     struct _ProcessList* waiting;
62 } Q3;
63
64 //cpu
65 typedef struct _CPU {
66     int preemptionTime; //used for Process from Q0 and Q1
67     int runningTime; //timer for cpu to compare with preemptionTime
68     int originQ; //origin of process to determine its next queue
69     Process* running; //process that occupies cpu now
70 } CPU;

```


(2) 함수

```
71
72 //reads input file and convert it into series of process
73 //containing corresponding values
74 //and put all process into one process list
75 //to be allocated to each RQ
76 void init(FILE* f, ProcessList* plist);
77
78 //Makes ProcessList
79 ProcessList* makeProcessList();
80
81 //Makes Process using input stream from the text file
82 Process* makeProcess(char* chars);
83
84 //Makes Linked list for burst time
85 BList* makeBList();
86
87 //put burst time into the burst time list
88 void insert2BList(BList* blist, int burst);
89
90 //put process into the process list
91 void insert2PList(ProcessList* plist, Process* p);
92
93 //un-link current element in the Process list
94 void segregate(ProcessList* plist);
95
96 //Increase waiting time for all process that are in the RQ
97 //and has not occupied cpu
98 void incrementWT(ProcessList* waiting);
99
100 Q0* makeQ0(); //Makes Q0. TQ set to two and wait-list generated
101 Q1* makeQ1(); //Makes Q1. TQ set to six and wait-list generated
102 Q2* makeQ2(); //Makes Q2. Shortest Process ptr and wait-list generated
103 Q3* makeQ3(); //Makes Q3. wait-list generated
104 CPU* makeCPU(); //Makes cpu.
105
106 //run function combines all functions below except for the last function
107 void run(ProcessList* plist, Q0* q0, Q1* q1, Q2* q2,
108         Q3* q3, CPU* mycpu, ProcessList* asleepList, ProcessList* zombieList);
109
110 //1) allocate processes whose arrival time <= Program time cycle
111 void allocate(int time, ProcessList* plist, Q0* q0, Q1* q1, Q2* q2, Q3* q3);
112
113 //2) determine which process will occupy cpu according to q0>q1>q2>q3 order
114 void Q0ProcessCheck(int time, Q0* q0, CPU* mycpu);
115 void Q1ProcessCheck(int time, Q1* q1, CPU* mycpu);
116 void Q2ShortestOneCheck(int time, Q2* q2, Q3* q3, CPU* mycpu);
117 void Q3ProcessCheck(int time, Q3* q3, CPU* mycpu);
```

```


119 //3) decrements I/O burst time and if I/O burst time is 0
120 //send the process the corresponding RQ
121 void IOInterruptCheck(int time, ProcessList* asleepList, Q0* q0, Q1* q1, Q2* q2);
122
123 //4) decrements cpu burst time of process that has occupied cpu
124 void runCPU(CPU* mycpu);
125
126 //5) if cpu burst time is 0, send the process to asleepList or zombieList
127 void goSleepOrOut(int time, CPU* mycpu, ProcessList* asleepList,
128     ProcessList* zombieList);
129
130 //6) after consuming time, check for preemption for Q0 and Q1
131 //if preemption occurs, q0->q1 waiting list, and q1->q2 waiting list
132 void preemptionCheckForQ01(int time, Q1* q1, Q2* q2, CPU* mycpu);
133
134 //show reports of process TT, WT, average TT, and average WT
135 void showReports(ProcessList* zombieList);
136
137 #endif

```

3 결과 분석

본 보고서에서는 테스트 파일 1,2,3의 결과를 txt파일, 결과 리포트, Gantt Chart(테스트 1은 텍스트, 2와 3은 그래픽으로 나타내고, 테스트 파일 4,5는 txt파일과 결과 reports로 나타낸다. 테스트파일 1,2,3,4,5의 txt파일, gcc환경에서 실행한 Detailed Reports는 첨부파일로 제공한다.

1) 테스트 파일 1

 input1 - Notepad

File Edit Format View Help

```

4
1      0      0      2      4 6 8
2      1      1      2      1 3 5
3      2      2      2      2 4 6
4      3      3      2      9 7 5

```

<그림3-1. Input1.txt>

```

***** Process reports *****
-----
Process 2's TT: 11      WT: 2
Process 3's TT: 19      WT: 6
Process 1's TT: 25      WT: 8
Process 4's TT: 46      WT: 23
-----
Process Count: 4
Average TT: 25.25      Average WT: 9.75
-----

```

<그림3-2. Input1의 결과 리포트>

cycle 0 starting!		occupy cpu!
process 1 going into q0 waiting q!	cycle 6 starting!	Process 1 burst!
Process 1 in Q0 is about to	Process 3 burst!	cycle 12 ended!
occupy cpu!	Process 3 Cpu Burst Time is over!	
Process 1 burst!	might sleep!	cycle 13 starting!
cycle 0 ended!	cycle 6 ended!	Process 1 burst!
		Process 1 scheduled from Q0 has
		been Preempted!
cycle 1 starting!	cycle 7 starting!	cycle 13 ended!
process 2 going into q1 waiting q!	Process 2 in Q0 is about to	
Process 1 burst!	occupy cpu!	
Process 1 scheduled from Q0 has	Process 2 burst!	cycle 14 starting!
been Preempted!	cycle 7 ended!	Process 3 in Q1 is about to
cycle 1 ended!		occupy cpu!
	cycle 8 starting!	Process 3 burst!
cycle 2 starting!	Process 2 burst!	cycle 14 ended!
process 3 going into q2 waiting q!	Process 2 scheduled from Q0 has	
Process 2 in Q1 is about to	been Preempted!	cycle 15 starting!
occupy cpu!	cycle 8 ended!	Process 3 burst!
Process 2 burst!		cycle 15 ended!
Process 2 Cpu Burst Time is over!	cycle 9 starting!	
might sleep!	Process 2 in Q1 is about to	cycle 16 starting!
cycle 2 ended!	occupy cpu!	Process 3 burst!
	Process 2 burst!	cycle 16 ended!
cycle 3 starting!	cycle 9 ended!	
process 4 going into q3 waiting q!		cycle 17 starting!
Process 1 in Q1 is about to	cycle 10 starting!	Process 3 burst!
occupy cpu!	Process 1 I/O burst time ended!	cycle 17 ended!
Process 1 burst!	going to wake up!	
cycle 3 ended!	Process 3 I/O burst time ended!	cycle 18 starting!
	going to wake up!	Process 3 burst!
cycle 4 starting!	Process 2 burst!	cycle 18 ended!
Process 1 burst!	cycle 10 ended!	
Process 1 Cpu Burst Time is over!		cycle 19 starting!
might sleep!	cycle 11 starting!	Process 3 burst!
cycle 4 ended!	Process 2 burst!	Process 3 Cpu Burst Time is over!
	Process 2 Cpu Burst Time is over!	might sleep!
cycle 5 starting!	might sleep!	Process 3 ended! going to zombie
Process 3 from Q2 is going to	Process 2 ended! going to zombie	List!
take cpu	List!	cycle 19 ended!
Process 2 I/O burst time ended!	cycle 11 ended!	
going to wake up!		cycle 20 starting!
Process 3 burst!	cycle 12 starting!	Process 1 in Q1 is about to
cycle 5 ended!	Process 1 in Q0 is about to	occupy cpu!

<그림3-3. Input1결과(Text type Gantt chart built to be shown to user) 1/2>

Process 1 burst!	cycle 29 ended!	Process 4 I/O burst time ended!
cycle 20 ended!		going to wake up!
	cycle 30 starting!	cycle 41 ended!
cycle 21 starting!	Process 4 burst!	
Process 1 burst!	cycle 30 ended!	cycle 42 starting!
cycle 21 ended!		Process 4 from Q2 is going to
	cycle 31 starting!	take cpu
cycle 22 starting!	Process 4 burst!	Process 4 burst!
Process 1 burst!	cycle 31 ended!	cycle 42 ended!
cycle 22 ended!		
	cycle 32 starting!	cycle 43 starting!
cycle 23 starting!	Process 4 burst!	Process 4 burst!
Process 1 burst!	cycle 32 ended!	cycle 43 ended!
cycle 23 ended!		
	cycle 33 starting!	cycle 44 starting!
cycle 24 starting!	Process 4 burst!	Process 4 burst!
Process 1 burst!	cycle 33 ended!	cycle 44 ended!
cycle 24 ended!		
	cycle 34 starting!	cycle 45 starting!
cycle 25 starting!	Process 4 burst!	Process 4 burst!
Process 1 burst!	Process 4 Cpu Burst Time is over!	cycle 45 ended!
Process 1 Cpu Burst Time is over!	might sleep!	
might sleep!	cycle 34 ended!	cycle 46 starting!
Process 1 ended! going to zombie		Process 4 burst!
List!	cycle 35 starting!	Process 4 Cpu Burst Time is over!
cycle 25 ended!	cycle 35 ended!	might sleep!
		Process 4 ended! going to zombie
cycle 26 starting!	cycle 36 starting!	List!
Process 4 in Q3 is about to	cycle 36 ended!	cycle 46 ended!
occupy cpu!		
Process 4 burst!	cycle 37 starting!	
cycle 26 ended!	cycle 37 ended!	
cycle 27 starting!	cycle 38 starting!	
Process 4 burst!	cycle 38 ended!	
cycle 27 ended!		
	cycle 39 starting!	
cycle 28 starting!	cycle 39 ended!	
Process 4 burst!		
cycle 28 ended!	cycle 40 starting!	
	cycle 40 ended!	
cycle 29 starting!		
Process 4 burst!	cycle 41 starting!	

<그림3-4. Input1결과(Text type Gantt chart built to be shown to user) 2/2>

2) 테스트 파일 2

input2 - Notepad

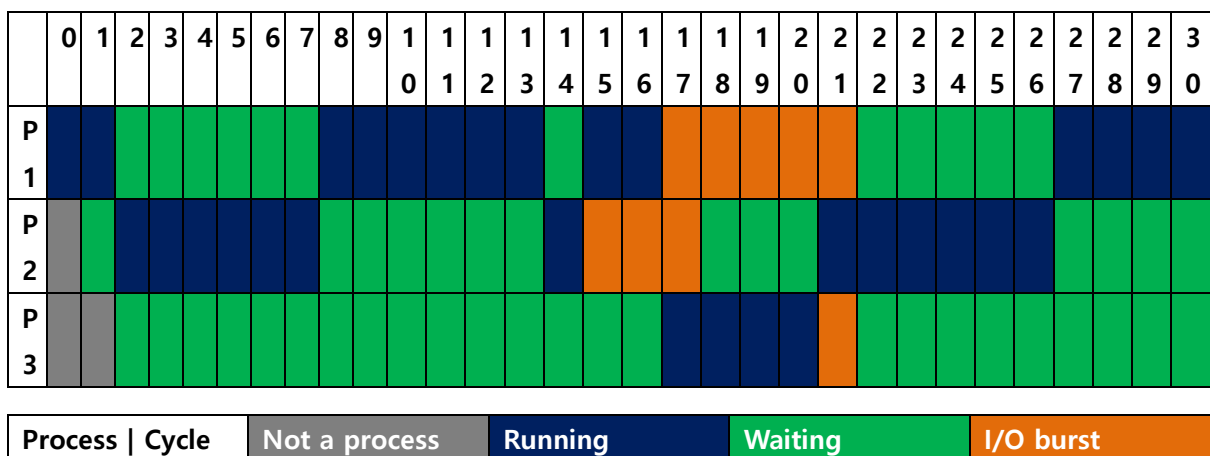
File Edit Format View Help

```
3
1      0      0      3      10 5 10 5 10
2      1      1      3      7 3 7 7 3
3      2      2      3      4 1 4 1 4
```

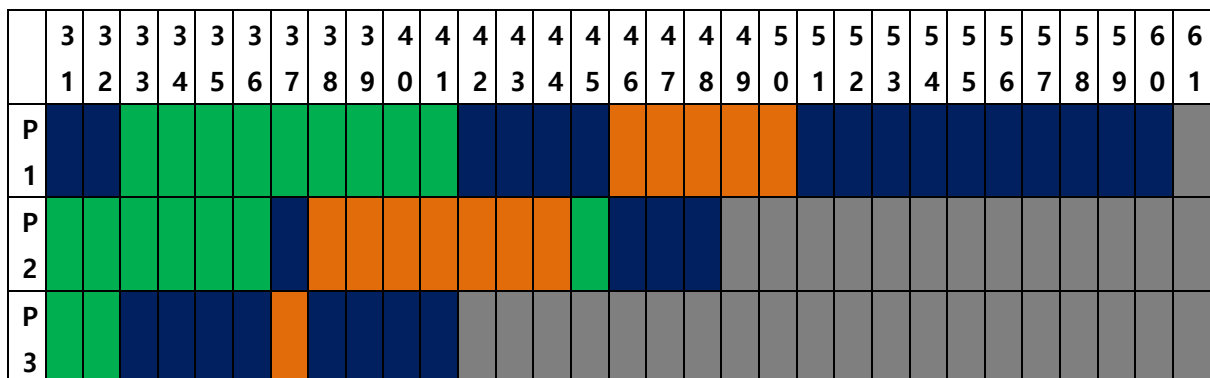
<그림3-5. Input2.txt>

```
***** Process reports *****
-----
Process 3's TT: 41      WT: 26
Process 2's TT: 48      WT: 21
Process 1's TT: 60      WT: 21
-----
Process Count: 3
Average TT: 49.67      Average WT: 22.67
-----
```

<그림3-6. Input2의 결과 리포트>



<그림 3-7. Input2의 Gantt Chart(txt파일을 표 형식으로 변환) 1/2>



<그림 3-8. Input2의 Gantt Chart(txt파일을 표 형식으로 변환) 2/2>

3) 테스트 파일 3

input3 - Notepad

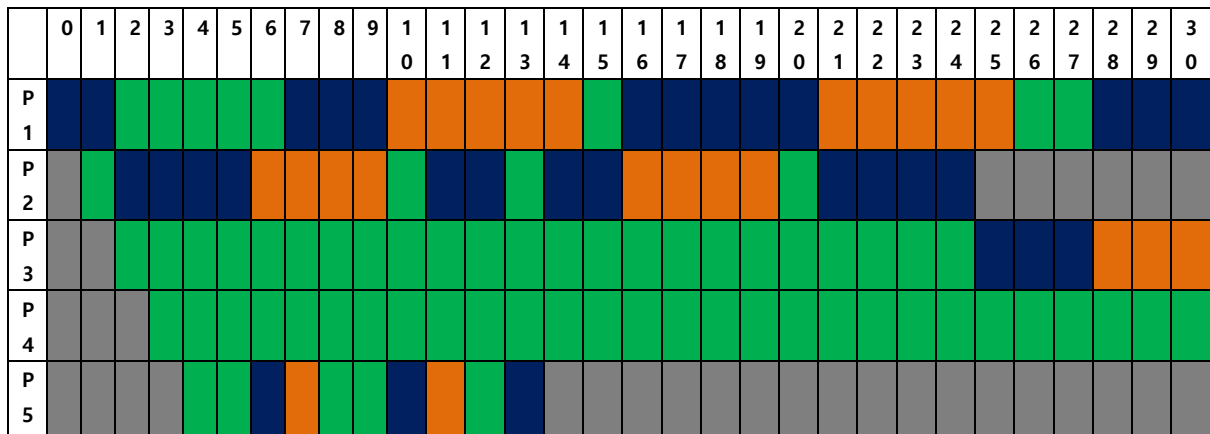
File Edit Format View Help

```
5
1      0      0      3      5 5 5 5 5
2      1      1      3      4 4 4 4 4
3      2      2      3      3 3 3 3 3
4      3      3      3      2 2 2 2 2
5      0      4      3      1 1 1 1 1
```

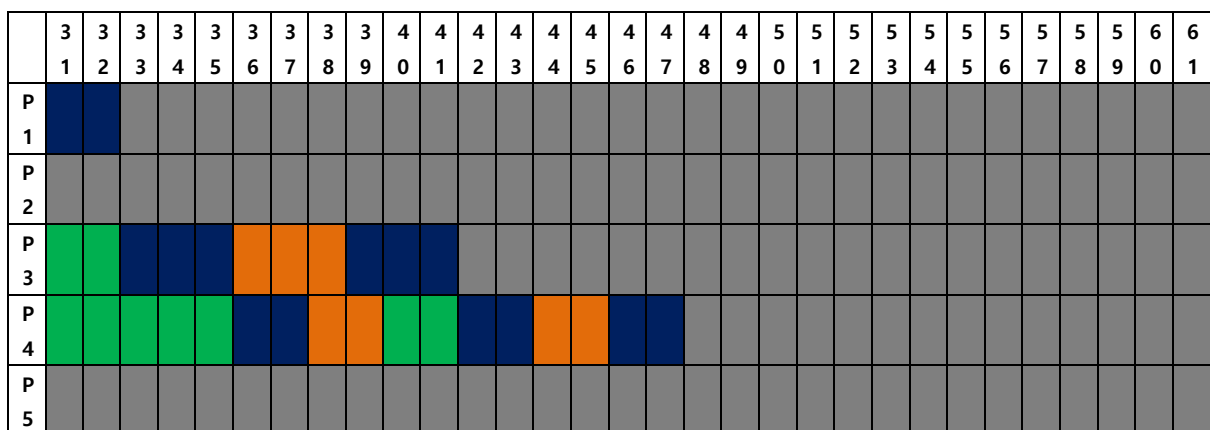
<그림3-9. Input3.txt>

```
***** Process reports *****
-----
Process 5's TT: 13      WT: 5
Process 2's TT: 24      WT: 4
Process 1's TT: 32      WT: 8
Process 3's TT: 41      WT: 25
Process 4's TT: 47      WT: 35
-----
Process Count: 5
Average TT: 31.40      Average WT: 15.40
-----
```

<그림3-10. Input3의 결과 리포트>



<그림 3-11, 3-12. Input3의 Gantt Chart(txt파일을 표 형식으로 변환) 1/2, 2/2>



4) 테스트 파일 4

input4 - Notepad

File Edit Format View Help

```

15
1      0      0      3      16 6 30 21 16
2      1      1      3      10 8 8 9 16
3      2      2      3      13 14 10 8 10
4      3      2      3      14 15 10 16 10
5      0      0      5      20 18 15 10 20 15 16 18 16
6      1      2      6      55 15 40 20 45 18 50 20 40 15 50
7      2      4      10     13 26 37 43 52 69 70 80 90 10 11 12 13 14 15 16 17 18 19
8      3      6      7      10 5 10 5 10 5 10 5 10 5 10
9      0      8      7      10 5 10 5 10 5 10 5 10 5 10
10     1     10      7      10 5 10 5 10 5 10 5 10 5 10
11     2     12      7      10 5 10 5 10 5 10 5 10 5 10
12     3     14      7      10 5 10 5 10 5 10 5 10 5 10
13     0     16      7      10 5 10 5 10 5 10 5 10 5 10
14     1     18      7      10 5 10 5 10 5 10 5 10 5 10
15     2     20     10     1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

```

<그림3-13. Input4.txt>

```

***** Process reports *****
-----
Process 15's TT: 101      WT: 27
Process 9's TT: 311      WT: 55
Process 13's TT: 327     WT: 67
Process 10's TT: 331     WT: 54
Process 14's TT: 335     WT: 57
Process 2's TT: 379      WT: 12
Process 3's TT: 491      WT: 7
Process 11's TT: 495     WT: 32
Process 1's TT: 584      WT: 15
Process 5's TT: 631      WT: 81
Process 4's TT: 887      WT: 810
Process 8's TT: 1081     WT: 947
Process 6's TT: 1167     WT: 126
Process 12's TT: 1302    WT: 1188
Process 7's TT: 1416     WT: 41
-----
Process Count: 15
Average TT: 655.87      Average WT: 234.60
-----

```

<그림3-14.Input4.txt의 결과 리포트>

5) 테스트 파일 5

input5 - Notepad

File Edit Format View Help

```

25
1      3      0      1      1
2      1      0      2      1 2 3
3      2      0      3      1 2 3 4 5
4      0      1      4      1 2 3 4 5 6 7
5      1      1      5      1 2 3 4 5 6 7 8 9
6      3      2      6      1 2 3 4 5 6 7 8 9 10 11
7      2      3      7      1 2 3 4 5 6 7 8 9 10 11 12 13
8      0      5      8      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
9      3      8      9      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
10     1     13     10     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
11     0     21     11     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
12     2     34     12     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
13     1     55     13     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
14     3     89     14     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
15     1    144     15     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
16     2    233     16     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
17     3    377     17     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
18     0    610     18     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
19     1    987     19     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
20     2   1588     20     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
21     1   2581     21     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
22     3   3995     22     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
23     0   6000     23     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
24     3   9100     24     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
25     2  12000     25     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

```

<그림3-15. Input5.txt>

```

***** Process reports *****
-----
Process 2's TT: 10      WT: 5
Process 4's TT: 51      WT: 23
Process 3's TT: 161     WT: 15
Process 5's TT: 171     WT: 47
Process 7's TT: 436     WT: 39
Process 8's TT: 484     WT: 89
Process 10's TT: 720    WT: 73
Process 11's TT: 905    WT: 89
Process 12's TT: 1608   WT: 86
Process 13's TT: 1782   WT: 91
Process 15's TT: 2169   WT: 119
Process 16's TT: 2368   WT: 129
Process 18's TT: 2723   WT: 171
Process 20's TT: 3044   WT: 192
Process 21's TT: 3190   WT: 263
Process 1's TT: 3273    WT: 3273
Process 6's TT: 3420    WT: 3320
Process 23's TT: 3538   WT: 308
Process 9's TT: 3636    WT: 3413
Process 14's TT: 4281   WT: 3387
Process 17's TT: 4569   WT: 3396
Process 25's TT: 4795   WT: 154
Process 22's TT: 5174   WT: 2982
Process 24's TT: 5424   WT: 2745
Process 19's TT: 5662   WT: 3912
-----
Process Count: 25
Average TT: 2543.76      Average WT: 1132.84
-----

```

<그림3-16. Input5의 결과 리포트>