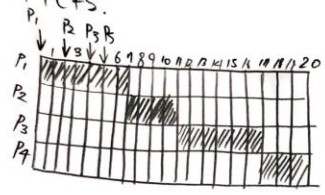
(15)1. 다음의 각 CPU 스케줄링 알고리즘들에 대해 프로세스들의 평균 대기 시간(average waiting time)을 계산하시오. 아래 표에서 프로세스들의 도착시간이 갖는 의미는, 프로세스가 도착시 해당 프로세스가 먼저 준비큐에 저장된 다음, 스케줄링이 이루어진다는 뜻이다. 간트 차트와 계산 과정이 답안에 명시되어야 한다.

프로세스	CPU 시간	도착시간
P1	7	0.0
P2	4	2.0
P3	6	4.0
P4	3	5.0

①FCFS(First-Come-First-Served)
②SPN(Shortest Process Next)
③SRT(Shortest-Remaining-Time)
④RR(Round-Robin): 시간 할당량 = 2
⑤HRRN(Highest Response Ratio Next)

20160860 강은구

1. FCFS.



t=0 — P1

t=2 — P2 | P1

t=4 — P3 | P2 | P1

t=5 — P4 | P3 | P2 | P1

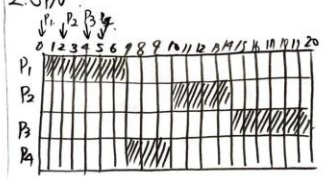
t=7 — P4 | P3 | P2

t=11 — P4 | P1

t=11 — P4

t=20 — P4

2. SPN.



t=0 — P1

t=2 — P2 | P1

t=4 — P3 | P2 | P1

t=5 — P4 | P3 | P2 | P1

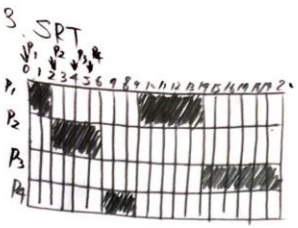
t=7 — P3 | P2 | P4

t=10 — P3 | P2

t=14 — P3

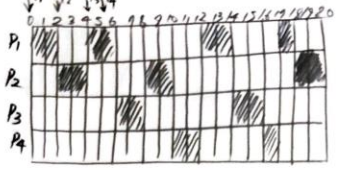
t=20 — P3

3. SPT



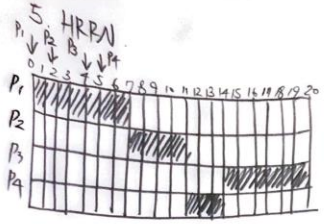
t=0 — P1
t=2 — P1 | P2
t=4 — P3 | P1 | P2
t=5 — P4 | P3 | P1 | P2
t=6 — P3 | P1 | P4
t=9 — P3 | P1
t=11 — P3
t=20 — P3

4. RR (Round Robin) q=2



t=0 — P1
t=2 — P1 | P2
t=4 — P2 | P3 | P1
t=5 — P4 | P2 | P3 | P1
t=6 — P1 | P4 | P2 | P3
t=8 — P3 | P1 | P4 | P2
t=10 — P3 | P1 | P4
t=12 — P4 | P3 | P1
t=14 — P1 | P4 | P3
t=16 — P3 | P4 | P4
t=17 — P3 | P1
t=18 — P3
t=20 — P3

5. HRRN



t=0 — P1
t=2 — P2 | P1
t=4 — P3 | P2 | P1
t=5 — P4 | P3 | P2 | P1

t=1 시점에서 각 프로세스의 Ratio는

$P_2 = \frac{5+4}{4} = \frac{9}{4}$

$P_3 = \frac{3+6}{6} = \frac{9}{6}$

$P_4 = \frac{2+3}{3} = \frac{5}{3}$

이므로 P2의 값이 가장 크다.

t=9 — P4 | P3 | P2

t=11 시점에서 각 프로세스의 Ratio는

$P_3 = \frac{9+6}{6} = \frac{15}{6} = \frac{5}{2}$

$P_4 = \frac{6+3}{3} = \frac{9}{3} = 3$

∴ P4의 값이 가장 크다.

t_{11} = P3 | P4

t_{14} — P3

t_{20} — P3

(10)2. 아래 문항에 대하여 설명하시오.

- 디렉토리 위치:~shcho/class/os/ProCon/competingThreads/TVContribution.java

①이 java 코드에서 공유 변수는 무엇인가?

②이 공유 변수에 대한 상호 배제가 어떻게 적용되고 있는지 설명하시오.

③실행 결과에 대하여 설명하시오.

```
-bash-3.2$ cat TVContribution.java
class Account {
  private int total = 0;
  synchronized void deposit(int amount) {
    total += amount;
  }
  int gettotal() {
    return total;
  }
}

class Customer extends Thread {
  Account account1;
  int thNum;

  Customer(Account account, int i) {
    this.account1 = account;
    thNum = i;
  }
  public void run() {
    try {
      for(int i = 0; i < 200 ; i++) {
        System.out.println("" + getName() + " : " + i +"");
        if ( (thNum % 2) == 0) {
                account1.deposit(1000);
                sleep(10);
        }
        else {
                account1.deposit(3000);
                sleep(2);
        }
        if (account1.gettotal() >= 100000) break;
      }
      catch(Exception e) {
        System.out.println(e);
      }
    }
}

class TVContribution {
  public static void main(String args[]) {
     Account account = new Account();

    Customer customers[] = new Customer[5];
    for(int i = 0; i < 5 ; i++) {
      customers[i] = new Customer(account, i);
      customers[i].start();
    }
    for(int i = 0; i < 5; i++) {
      try {
        customers[i].join();
      }
      catch(InterruptedException e) {
        e.printStackTrace();
      }
    }
    System.out.println(" : " + account.gettotal());
  }
}
```

1. 이 java 코드에서 공유 변수는 무엇인가?

- 클래스 Account의 total을 공유변수로 사용한다.

2. 이 공유 변수에 대한 상호 배제가 어떻게 적용되고 있는지 설명하시오.

- 분기에 의해Deposit 함수를 통하여 쓰레드를 실행하고 각기 다른 시간의 sleep함수를 사용하고 있다.

3. 실행 결과에 대하여 설명하시오.

thNum이 짝수이면 deposit함수를 통해1000원을 입금하고 10만큼 sleep한다. 만약 홀수이면 3000원을 입금하고 2만큼 sleep한다 만약 입금되는 총 금액이 100,000원 이상이 되면 break를 한다. 그리고 고객의 이름과 총 금액을 출력해준다.

(15)3. 특정 시점에서 한 시스템의 자원 할당 상태가 다음과 같다고 가정하자.

- 4개 프로세스: P1~P4
- 자원의 유형 4가지: R1~R4
- 총 자원량: R1(6개), R2(4개), R3(4개), R4(5개)

현 시점에서 자원 할당 스냅샷:

	R1	R2	R3	R4
P1	0	1	2	0
P2	1	1	0	1
P3	0	1	0	2
P4	2	0	2	0

<할당 행렬>

	R1	R2	R3	R4
P1	1	3	1	1
P2	1	0	1	1
P3	1	1	3	0
P4	2	0	0	1

<요청 행렬>

①이 컴퓨터 시스템내 P1~P4에게 할당된 자원들의 벡터는 어떻게 표현되는가?
②이 컴퓨터 시스템내 P1~P4가 요청하는 자원들의 벡터는 어떻게 표현되는가?
③현 상태에서 가용 벡터는 어떻게 표현되는가?
④현재 상태가 교착 상태인지 아닌지 설명하시오.

3. 2학년3반 강준구 ④

공유자원는 : R_1(6개), R_2(4개), R_3(4개), R_4(5개)

현시점에서 자원할당 스냅샷

	R_1	R_2	R_3	R_4
P_1	0	1	2	0
P_2	1	1	0	1
P_3	0	1	0	2
P_4	2	0	2	0

<할당 행렬>

	R_1	R_2	R_3	R_4
P_1	1	3	1	1
P_2	1	0	1	1
P_3	1	1	3	0
P_4	2	0	0	1

<요청 행렬>

① 할당벡터

R_1	R_2	R_3	R_4
3	3	4	3

② 요청벡터

R_1	R_2	R_3	R_4
5	4	5	3

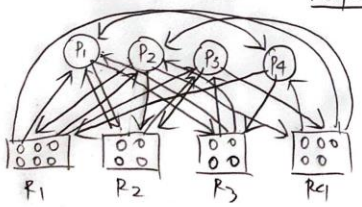
③ 가용벡터

R_1	R_2	R_3	R_4
3	1	0	2

④ 현재상태가 교착상태인지 아닌지 설명하시오.

	P_1	P_2	P_3	P_4	R_1	R_2	R_3	R_4
Step1	U	U	U	U	W	3	1	0	2
Step2	U	U	U	M	W	5	1	2	2
Step3	U	M	U	M	W	6	2	2	3

← Deadlock 이다. ∴ 교착상태이다



(10)4. 트랙 300개(0~299)로 구성된 디스크에 디스크 요청들이

112, 183, 35, 215, 87, 166, 51, 280, 98

의 순으로 디스크 요청 큐에 도착해 있고, 현재 디스크 헤드는 트랙 130에 위치하고 있다. SCAN와 C-SCAN 스케줄링 알고리즘에서는 트랙 번호가 낮은 방향으로 헤드가 이동한다고 가정한다. 다음 스케줄링 알고리즘에서 평균 디스크 헤드의 이동거리를 계산하시오. 계산기를 사용하는 것을 허용한다.

①FIFO ②SSTF(Shortest Seek Time First)

③SCAN ④C-SCAN(Circular SCAN)

4 현재 디스크헤드 130　　　20160136 강동구　　　⑤

큐 112, 183, 35, 215, 87, 166, 51, 280, 98

① FIFO

	이동거리
130	
112	18
183	71
35	148
215	180
87	128
166	79
51	115
280	229
98	182

평균이동거리 = 127.778.

② SSTF(Shortest Seek Time First)

130	
112	18
98	14
87	11
51	36
35	16
166	131
183	17
215	32
280	65.

평균이동거리 37.778.

③ SCAN 낮은방향으로 이동.

130	
112	18
98	14
87	11
51	36
35	16
166	131
183	17
215	32
280	65

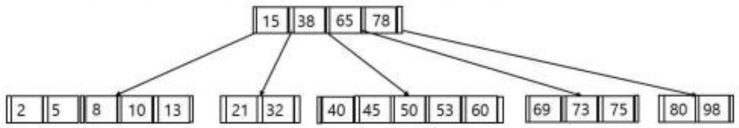
평균이동거리 37.778.

④ C-SCAN

130	
112	18
98	14
87	11
51	36
35	16
280	245
215	65
183	32
166	17

평균이동거리 50.44

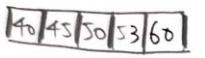
(10)5. 아래 제시된 트리에서 최소도(minimum degree)는 d=3이다. 이 트리에서 각 노도는 최대 (2d−1) 키를 갖고, 2d개 자식 노드를 가질 수 있다.



① 위 트리에 Key=58를 삽입하는 경우, 이 트리가 어떻게 되는지 설명하시오.
② 그 다음 Key=6를 삽입하는 경우, 이 트리가 어떻게 되는지 설명하시오.
③ 이러한 트리를 무엇이라고 하는가?
④ 이러한 트리의 장점은 무엇인지 설명하시오.

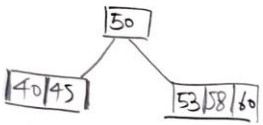
2016013G0 강윤구

5. 키트리에 5을 삽입하는 경우.

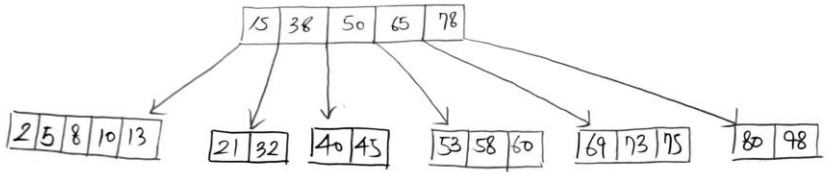


↑ 이자리에 위치해야한다.

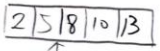
하지만 각 노드의 최대 5개의 키를 가지므로. 다음과 같이 쪼갠다.



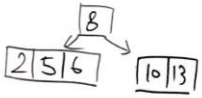
따라서



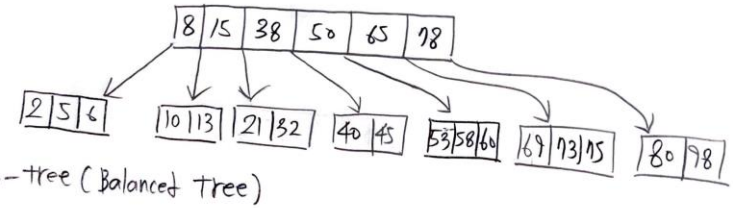
2) 그 다음 key= 6을 삽입하는 경우, 이트리가 어떻게 되는지 설명하시오.



↑ 이 자리에 위치해야 하지만 각 노드의 최대 5개의 키를 가지므로 다음과 같이 쪼개진다.



따라서



3) B-tree (Balanced tree)

4) -Leaves 노드들이 항상 똑같은 Level을 유지한다.

-Skewed tree가 발생하지 않는다.

-균등한 탐색속도가 보장된다.

(10)6. 아래 디렉토리를 참조하여 실행 결과 (a),(b)가 왜 그렇게 출력되는지 이유와 차이점이 무엇인지을 설명하시오.

```
- 디렉토리 위치:/home/shcho/class/os/io/
```

(a)

```
-bash-3.2$ a.out
abc
nread= 4, buf= abc
buf[0]=97, buf[1]=98, buf[2]=99, buf[3]=10, buf[4]=0,
^C
```

(b)

```
 -bash-3.2$ a.out
abc
nread= 4, smallbuf= abc
buf[0]=97 buf[1]=98 buf[2]=99 buf[3]=0
^C
```

(a)의 경우 'abc '를 입력했다. 따라서 첫 번째 들어온 buf[0]에는 a 의 아스키코드값 97 을 출력하며 두 번째 입력값 but[1]에는 b 가 들어와 98 을 출력해주었으며 세 번쨰 입력값 buf[2]에는 c 가 들어와 99 를 출력해주었다. 네 번째 입력값인 buf[3]에는 다음줄로 넘어가는 Line Feed 값이 들어왔으므로 이에 해당하는 아스키코드값 10 을 출력한다. 마지막 buf[4]에는 널값이 들어오므로 0 을 출력해준다.

(b)의 경우 'abc'를 입력했다. 따라서 첫 번째 들어온 buf[0]에는 a 의 아스키코드값 97 을 출력하며 두 번째 입력값 but[1]에는 b 가 들어와 98 을 출력해주었으며 세 번쨰 입력값 buf[2]에는 c 가 들어와 99 를 출력해주었다. 마지막 buf[4]에는 널값이 들어오므로 0 을 출력해준다.

(20)7. 아래 문항에 대하여 설명하시오.

- 디렉토리 위치:~shcho/class/os/ProCon/ProCon_mesg_unix

- 아래의 (a)는 etest가 4번 반복해서 먼저 실행한 후 (b)와 같이 실행한 것이다.
- 주의사항: 아래의 QKEY값이 동일하면 사용자들간에 충돌이 생겨 에러가 발생한다. QKEY값을 자신만의 고유한 수치로 바꿔 실행해야 에러가 발생하지 않는다.

```
#define QKEY   (key_t)0777
```

①실행 결과 (a)(b)에 대하여 설명하시오.

②msgget, msgsnd, msgrcv 함수들의 기능에 대하여 설명하시오.

③아래의 etest는 한번 한번씩 메시지를 송신하고 있다. serve처럼 계속 살아 있으면서 메시지를 반복적으로 보낼 수 있도록 개선하고 실행 결과를 화면 캡춰하여 제시한다. 개선된 소스 코드 이름은 "fin_학번_etest.c"으로 부여하고 (c)에서 지정된 디렉토리에 제출한다.

(a)
```
-bash-3.2$ etest message4 4
-bash-3.2$ etest message2 2
-bash-3.2$ etest message3 3
-bash-3.2$ etest message1 1
```

(b)
```
-bash-3.2$ serve
server process pid is 28981
 priority: 1 name: message1
 priority: 2 name: message2
 priority: 3 name: message3
 priority: 4 name: message4
```

(c)
```
작성한 소스 코드 os_fin_etest_자신의학번.c는 아래에 지정된 디렉토리로 제출한다.
- 소스 코드명 예시:os_fin_etest_20161111.c
- 제출 디렉토리:/home/shcho/21-2/OS/fin/No7
```

1. (a)의 경우 그대로 파일을 실행한 것이며 (b)의 경우 pid값을 생성하여 우선순위에 따라 메시지를 출력하고 있다.

2. msgget : key 인자값에 연관된 메시지큐의 식별번호를 반환한다.

- msgsnd : message queue로 데이터를 전송하는 함수이다.
- msgrcv : 메시지 큐로부터 데이터를 수신하는 함수이다.

```
-bash-3.2$ cat serve.c
#include"q.h"
int proc_obj(struct q_entry *);
int serve(void);
main()
{
        pid_t pid;
        switch(pid = fork()){
                case 0: /* ÀÚ½Ä */
                        serve();
                        break;
                case -1:
                        perror("fork to start server failed");
                        break;
                default:
                        printf("server process pid is %d\n", pid);
        }
        exit(pid != -1 ? 0 : 1);
}

int serve(void)
{
        int mlen, r_qid;
        struct q_entry r_entry;
if((r_qid = init_queue()) == -1)
                return (-1);

        for ( ; ; )        {
                if((mlen = msgrcv(r_qid, &r_entry, MAXOBN, /* 0,
*/ (-1 * MAXPRIOR), MSG_NOERROR)) == -1)
                {
                        perror("msgrcv failed");
                        return(-1);
                }
                else
                {
                        r_entry.mtext[mlen] = '\0';
                        proc_obj(&r_entry);
                  }        }}
int proc_obj(struct q_entry *msg)
{
```

```
        printf("\n priority: %ld name: %s \n", msg->mtype, msg-
>mtext);
}
```

```
-bash-3.2$ cat etest.c
#include<stdlib.h>
#include"q.h"

int enter(char *, int );

main(int argc, char **argv)
{
        int priority;

        if(argc !=3)
        {
                fprintf(stderr, "usage: %s objname prioritey\n",
argv[0]);
                exit(1);
        }

        if((priority = atoi(argv[2])) <= 0 || priority > MAXPRIOR)
        {
                perror("invalid priority");
                exit(2);
        }

        if(enter(argv[1], priority) < 0)
        {
                perror("enter failure");
                exit(3);
        }
        exit(0);
}

int enter(char *objname, int priority)
{
        int len, s_qid;
        struct q_entry s_entry;  /* [illegible] */
        /* [illegible], [illegible] */
        if((len = strlen(objname)) > MAXOBN)
        {
                perror("name too long");
                return(-1);
        }

        if(priority > MAXPRIOR || priority < 0)
              {
                perror("invalid priority level");
                return(-1);
        }

        /* [illegible] */
        if((s_qid = init_queue()) == -1)
                return(-1);

        /* s_entry */
        s_entry.mtype = (long) priority;
        strncpy(s_entry.mtext, objname, MAXOBN);

        /* [illegible] */
        if(msgsnd(s_qid, &s_entry, len, 0) == -1)
       {
                perror("msgsnd failed");
                return(-1);
        }
        else
                return(0);
}
-bash-3.2$
```

```
-bash-3.2$ cat q.h
#include<stdio.h>
#include<sys/ipc.h>
#include<string.h>
#include<sys/msg.h>
#include<sys/types.h>

#define QKEY   (key_t)0721  /* [illegible] */
```

```c
#define QPERM  0660          /* ▒▒ ▒▒ */
#define MAXOBN 50            /* ▒▒ ▒▒ ▒▒▒▒ */
#define MAXPRIOR 10            /* ▒▒ ▒▒ ▒▒▒ ▒▒▒ */

int init_queue(void);

struct q_entry{
        long mtype;
        char mtext[MAXOBN+1];
};

int init_queue(void)
{
        int queue_id;

        /* ▒▒▒ ▒▒ ▒▒▒ ▒▒▒▒ ▒▒▒▒▒ ▒▒ ▒▒. */
        if((queue_id = msgget(QKEY, IPC_CREAT | QPERM)) == -1)
                perror("msgget failed");

        return(queue_id);
};
```