



System Programming Assignment

▼ 과제 1

- 구조체를 사용해서 Linked List로 계산 노드(add/sub/mul/div)기능 구현하기
- 현재까지 생성된 계산 노드 출력
- 현재까지 생성된 계산 노드들의 계산 결과를 출력
- 전체 삭제
- 프로그램 종료

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

struct CalculatorNode* startLocation = NULL;

struct CalculatorNode {
    float a;
    float b;
    char name[20];
    int(*fptr)(int, int);
    struct CalculatorNode* next; //다음 노드의 주소를 저장할 포인터
};

int add(int, int);
int sub(int, int);
int mul(int, int);
int divs(int, int);
```

```

int add(int a, int b)
{
    int result = a + b;
    return result;
}
int sub(int a, int b)
{
    int result = a - b;
    return result;
}
int mul(int a, int b)
{
    int result = a * b;
    return result;
}
int divs(int a, int b)
{
    float result = (float)a / (float)b;
    return result;
}

int main()
{
    while (1) {
        printf("Choose->1.노드생성 2.노드모두출력 3.노드결과모두출력 4.노드모두삭제 5.종료:\n");

        int i;
        scanf("%d", &i);

        if (i == 5)
        {
            printf("종료합니다");
            break;
        }
        switch (i)
        {
        case 1:
            printf("노드 생성 모드입니다. 숫자 2개 입력\n");

            float a;
            float b;
            char sign;
            printf("Enter the num1:");
            scanf("%f", &a);
            printf("Enter the num2:");
            scanf("%f", &b);
            printf("Enter the Calculator sing(e.g. '+', '-', '*', '/'):");
            scanf("\n%c", &sign);
            struct CalculatorNode* node = malloc(sizeof(struct CalculatorNode));
            node->a = a;
            node->b = b;
            node->next = NULL;

            int(*fptr)(int, int)=NULL;
            int x;

            if (sign == '+')
            {
                fptr = add;
                x = fptr(a, b);
                printf("%d\n", x);
            }
            else if (sign == '-')
            {
                fptr = sub;
                x = fptr(a, b);
                printf("%d\n", x);
            }
            else if (sign == '*')
            {
                fptr = mul;
                x = fptr(a, b);
                printf("%d\n", x);
            }
            else if (sign == '/')
            {
                fptr = divs;
                x = fptr(a, b);
                printf("%d\n", x);
            }
        }
    }
}

```

```

    }
    if (sign == '-')
    {
        fptr = sub;
        x = fptr(a, b);
        printf("%d\n", x);
    }
    if (sign == '*')
    {
        fptr = mul;
        x = fptr(a, b);
        printf("%d\n", x);
    }
    if (sign == '/')
    {
        fptr = divs;
        x = fptr(a, b);
        printf("%d\n", x);
    }

    node->fptr = fptr;

    if (startLocation == NULL) {
        startLocation = node;
    }

    else
    {
        struct CalculatorNode* temp = startLocation;
        if (temp == NULL) {
            startLocation = temp;
        }
        else {
            while (1)
            {
                if (temp->next == NULL)
                {
                    temp->next = node;
                    break;
                }

                else
                    temp = temp->next;
            }
        }
        printf("노드를 생성하고 추가했습니다.");
    }

    break;
case 2:
    if (startLocation == NULL)
    {
        printf("노드가 하나도 없음\n");
    }
    else
    {
        struct CalculatorNode* temp = startLocation;
        while (1)
        {
            if (temp->next == NULL)
                break;
            else

```

```

        temp = temp->next;
    }
}

break;
case 3:
if (startLocation == NULL)
{
    printf("노드가 하나도 없음\n");
}
else
{
    struct CalculatorNode* temp = startLocation;
    while (1)
    {
        printf("%d\n", temp->fptr(temp->a, temp->b));
        if (temp->next == NULL)
        {
            break;
        }
        else
            temp = temp->next;
    }
}

break;
case 4:
if (startLocation == NULL)
{
    printf("노드가 하나도 없습니다.\n");
}
else
{
    struct CalculatorNode* temp = startLocation;
    while (1)
    {
        struct CalculatorNode* temp2 = temp->next;
        free(temp);
        if (temp2 == NULL)
            break;
        else
            temp = temp2;

    }
}

break;
case 5:
break;
}
}
}

```

▼ 과제 2

출력 결과를 ls2형식으로 보여주되 다음의 형식을 따를 것.

- 내가 현재 있는 디렉토리의 전체 파일의 개수 출력

- 내가 현재 있는 디렉토리의 전체 파일의 크기의 합을 출력
 - “./ls3 -t dir” : dir 디렉토리를 시간 순으로 출력하기
 - “./ls3 -b dir” : dir 디렉토리를 파일의 크기 순으로 출력하기
- ls2 형식으로 출력한다는 것은 아래와 같다.

```
kimgrace@Kimgrace:~/lab4$ ./ls2
-rw-r--r-- 1kimgracekimgrace12288 Sep 22 11:31 ls2.c.swp
-rw-r--r-- 1kimgracekimgrace12288 Sep 27 13:09 ls2.c.swj
-rw-r--r-- 1kimgracekimgrace4512 Oct 26 19:44 ls3.c
-rw-r--r-- 1kimgracekimgrace12288 Sep 27 13:04 ls2.c.swn
-rw-r--r-- 1kimgracekimgrace12288 Sep 27 13:05 ls2.c.swm
-rw-r--r-- 1kimgracekimgrace12288 Sep 27 13:06 ls2.c.swl
-rwxr-xr-z 1kimgracekimgrace21824 Oct 8 14:27 ls3
-rwxr-xr-z 1kimgracekimgrace16976 Sep 26 23:27 ls1
-rw-r--r-- 1kimgracekimgrace12288 Sep 27 01:02 ls2.c.swo
-rwxr-xr-z 1kimgracekimgrace16937 Oct 4 21:20 fileinfo
-rw-r--r-- 1kimgracekimgrace3034 Oct 8 14:25 ls2.c
-rw-r--r-- 1kimgracekimgrace24576 Sep 27 13:08 ls2.c.swk
drwxr-xr-z 2kimgracekimgrace4096 Oct 4 22:03 ls3_2
-rwxr-xr-z 1kimgracekimgrace17656 Oct 8 14:25 ls2
drwxr-xr-z 18kimgracekimgrace4096 Oct 26 19:44 ..
drwxrwxr-z 3kimgracekimgrace4096 Oct 26 19:44 .
-rw-rw-r-- 1kimgracekimgrace522 Oct 26 18:51 ls1.c
-rw-r--r-- 1kimgracekimgrace598 Oct 4 21:20 fileinfo.c
전체 파일 개수: 18
전체 파일 크기: 192651
```

- 그리고 시간순과 파일 크기 순으로 출력하는 것은 각각 -t, -b 옵션을 통해 보여준다.
- 다른 것과 가장 큰 차이는 우선, 다른 디렉토리가 있다는 소리인데, ls3에 ls2에 -t, -b 옵션을 추가한 코드를 작성해서 ls3_2라는 디렉토리에 복사 해 줄 것이다.
- 그럼 ls2의 코드부터 살펴보자

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <dirent.h>
#include <sys/stat.h>
#include <pwd.h>
void do_ls(char[]);
void dostat(char*);
void show_file_info(char*, struct stat*);
void mode_to_letters(int, char[]);
char * uid_to_name(uid_t);
char * gid_to_name(gid_t);
int i;
int j = 0;
int main(int argc, char*argv[]){
    if(argc==1){
```

```

        do_ls(".");
    }
    else
    {
        while(--argc){
            printf("%s:\n", *++argv);
            do_ls(*argv);
        }
    }
}

void do_ls(char dirname[]){
    DIR *dir_ptr;
    struct dirent* direntp;
    if((dir_ptr = opendir(dirname)) == NULL)
        fprintf(stderr, "ls2: cannot open %s\n", dirname);
    else
    {
        while ((direntp = readdir(dir_ptr)) != NULL)
            dostat(direntp->d_name);
        closedir(dir_ptr);
    }
    printf("전체 파일 개수: %d\n", i);
    printf("전체 파일 크기: %d\n", j);
}
void dostat(char* filename){
    struct stat info;
    if (stat (filename, &info) == -1)
        perror(filename);
    else{
        show_file_info(filename, &info);
    }
}

void show_file_info(char * filename, struct stat*info_p)
{
    char *uid_to_name(), *ctime(), *gid_to_name(), *filemode();
    void mode_to_letters();
    char modestr[11];
    i = i + 1;
    j += (long)info_p->st_size;
    mode_to_letters(info_p->st_mode, modestr);
    printf("%s", modestr);
    printf("%4d", (int)info_p->st_nlink);
    printf("%-8s", uid_to_name(info_p->st_uid));
    printf("%-8s", gid_to_name(info_p->st_gid));
    printf("%-8ld", (long)info_p->st_size);
    printf("%.12s", 4+ctime(&info_p->st_mtime));
    printf("%s\n", filename);
}
void mode_to_letters(int mode, char str[]){
    strcpy(str, "-----");

    if(S_ISDIR(mode)) str[0] = 'd';
    if(S_ISCHR(mode)) str[0] = 'c';
    if(S_ISBLK(mode)) str[0] = 'b';

    if(mode & S_IRUSR) str[1] = 'r';
    if(mode & S_IWUSR) str[2] = 'w';
    if(mode & S_IXUSR) str[3] = 'x';
}

```

```

        if(mode & S_IRGRP) str[4] = 'r';
        if(mode & S_IWGRP) str[5] = 'w';
        if(mode & S_IXGRP) str[6] = 'x';

        if(mode & S_IROTH) str[7] = 'r';
        if(mode & S_IWOTH) str[8] = 'w';
        if(mode & S_IXOTH) str[9] = 'z';
    }

char *uid_to_name(uid_t uid){
    struct passwd *getpwuid(), *pw_ptr;
    static char numstr[10];

    if((pw_ptr = getpwuid(uid)) == NULL){
        sprintf(numstr, "%d", uid);
        return numstr;
    }
    else
        return pw_ptr->pw_name;
}

#include <grp.h>

char *gid_to_name(gid_t gid){
    struct group *getgrgid(), *grp_ptr;
    static char numstr[10];
    if((grp_ptr = getgrgid(gid)) == NULL)
    {
        sprintf(numstr, "%d", gid);
        return numstr;
    }
    else
        return grp_ptr->gr_name;
}

```

- 이제 ls3.c 코드를 살펴보자.

```

#include <dirent.h>
#include <grp.h>
#include <pwd.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <time.h>

void do_ls(char dirname[]);
void dostat(char *, struct stat *);
void show_file_info(char *, struct stat *);
void mode_to_letters(int, char[]);
char *uid_to_name(uid_t);
char *gid_to_name(gid_t);
void set_option_flag(char opts[]);

int i;
int j = 0;

// opt
// t: sort by time

```

```

// b: sort by file size
int opt_flag = 0;

int main(int argc, char *argv[]) {
    if (argc == 1) {
        do_ls(".");
    } else {
        int i = 0;
        for (i = 0; i < argc; i++) {
            char *temp;
            if (argv[i][0] == '-') {
                set_option_flag(argv[i]);
                temp = argv[i];
                argv[i] = argv[argc - 1];
                argv[argc - 1] = argv[i];
                argc--;
                i--;
            }
        }
        while (--argc) {
            printf("%s:\n", *++argv);
            do_ls(*argv);
        }
    }
}

void set_option_flag(char opts[]) {
    int i = 0;
    for (i = 0; i < strlen(opts); i++) {
        if (opts[i] == 'b') {
            opt_flag = opt_flag | 1;
        } else if (opts[i] == 't') {
            opt_flag = opt_flag | (1 << 1);
        }
    }
}

void do_ls(char dirname[]) {

    DIR *dir_ptr;
    struct dirent *direntp;
    struct stat dirinfos[100];
    struct dirent *direntps[100];
    int cnt_info = 0;
    int i = 0;
    if ((dir_ptr = opendir(dirname)) == NULL)
        fprintf(stderr, "ls2: cannot open %s\n", dirname);
    else {
        while ((direntp = readdir(dir_ptr)) != NULL) {
            char location[1024];
            sprintf(location, "%s/%s", dirname, direntp->d_name);

            direntps[cnt_info] = direntp;
            dostat(location, &dirinfos[cnt_info]);

            cnt_info++;
        }
        if ((opt_flag & 1) != 0) {
            // process -b option
            int i = 0, j = 0;

```

```

        for (i = cnt_info - 1; i > 0; i--) {
            for (j = 0; j < i; j++) {
                if (dirinfos[j].st_size > dirinfos[j + 1].st_size) {
                    struct stat tempDirinfo = dirinfos[j];
                    struct dirent *tempDirent = direntps[j];
                    dirinfos[j] = dirinfos[j + 1];
                    direntps[j] = direntps[j + 1];
                    dirinfos[j + 1] = tempDirinfo;
                    direntps[j + 1] = tempDirent;
                }
            }
        }

        if (opt_flag & (1 << 1)) {
            // process -t option
            int i = 0, j = 0;
            for (i = cnt_info - 1; i > 0; i--) {
                for (j = 0; j < i; j++) {
                    if (dirinfos[j].st_mtime > dirinfos[j + 1].st_mtime) {
                        struct stat tempDirinfo = dirinfos[j];
                        struct dirent *tempDirent = direntps[j];
                        dirinfos[j] = dirinfos[j + 1];
                        direntps[j] = direntps[j + 1];
                        dirinfos[j + 1] = tempDirinfo;
                        direntps[j + 1] = tempDirent;
                    }
                }
            }
        }

        for (i = 0; i < cnt_info; i++) {
            show_file_info(direntps[i]->d_name, &dirinfos[i]);
        }

        closedir(dir_ptr);
    }

    printf("전체 파일 개수: %d\n", i);
    printf("전체 파일 크기: %d\n", j);
}

void dostat(char *filename, struct stat *info) {
    if (stat(filename, info) == -1)
        perror(filename);
}

void show_file_info(char *filename, struct stat *info_p) {
    char modestr[11];
    i = i + 1;
    j += (long)info_p->st_size;
    mode_to_letters(info_p->st_mode, modestr);
    printf("%s", modestr);
    printf("%4d", (int)info_p->st_nlink);
    printf("%8s", uid_to_name(info_p->st_uid));
    printf("%8s", gid_to_name(info_p->st_gid));
    printf("%8ld", (long)info_p->st_size);
    printf(" %.12s", 4 + ctime(&info_p->st_mtime));
    printf(" %s\n", filename);
}

void mode_to_letters(int mode, char str[]) {
    strcpy(str, "-----");
}

```

```

if (S_ISDIR(mode))
    str[0] = 'd';
if (S_ISCHR(mode))
    str[0] = 'c';
if (S_ISBLK(mode))
str[0] = 'b';

if (mode & S_IRUSR)
    str[1] = 'r';
if (mode & S_IWUSR)
    str[2] = 'w';
if (mode & S_IXUSR)
    str[3] = 'x';

if (mode & S_IRGRP)
    str[4] = 'r';
if (mode & S_IWGRP)
    str[5] = 'w';
if (mode & S_IXGRP)
    str[6] = 'x';

if (mode & S_IROTH)
    str[7] = 'r';
if (mode & S_IWOTH)
    str[8] = 'w';
if (mode & S_IXOTH)
    str[9] = 'z';
}

char *uid_to_name(uid_t uid) {
    struct passwd *getpwuid(), *pw_ptr;
    static char numstr[10];

    if ((pw_ptr = getpwuid(uid)) == NULL) {
        sprintf(numstr, "%d", uid);
        return numstr;
    } else
        return pw_ptr->pw_name;
}

#include <grp.h>

char *gid_to_name(gid_t gid) {
struct group *getgrgid(), *grp_ptr;
    static char numstr[10];
    if ((grp_ptr = getgrgid(gid)) == NULL) {
        sprintf(numstr, "%d", gid);
        return numstr;
    } else
        return grp_ptr->gr_name;
}

```

- 이제 각각의 옵션들로 컴파일을 해 보면, 실행이 잘 될 것이다.

```

kimgrace@Kimgrace:~/lab4$ ./ls3 -t ls3_2
ls3_2:
drwxr-xr-z 2kimgracekimgrace 4096 Oct  4 22:03 .
-rw-r--r-- 1kimgracekimgrace 3050 Oct  4 22:05 ls3.c
drwxrwxr-z 3kimgracekimgrace 4096 Oct 26 20:03 ..
전체 파일 개수: 3
전체 파일 크기: 11242
kimgrace@Kimgrace:~/lab4$ ./ls3 -b ls3_2
ls3_2:
-rw-r--r-- 1kimgracekimgrace 3050 Oct  4 22:05 ls3.c
drwxrwxr-z 3kimgracekimgrace 4096 Oct 26 20:03 ..
drwxr-xr-z 2kimgracekimgrace 4096 Oct  4 22:03 .
전체 파일 개수: 3
전체 파일 크기: 11242

```



만약, ls3_2 파일에 복사되어있지 않는 파일로 실행시켜보면, (현재 ls3_2에는 ls3 파일 밖에 없음) 해당 파일을 -t로 열 수 없다고 뜨게 된다.

```

현재 디렉토리: 192.168.1.10~2051
kimgrace@Kimgrace:~/lab4$ ./ls2 -t ls3_2
-t:
ls2: cannot open -t
전체 파일 개수: 0
전체 파일 크기: 0
ls3_2:
-rw-r--r-- 1kimgracekimgrace4512      Oct 26 19:44ls3.c
drwxr-xr-z 18kimgracekimgrace4096      Oct 26 20:03..
drwxrwxr-z 3kimgracekimgrace4096      Oct 26 20:03.
전체 파일 개수: 3
전체 파일 크기: 12704
kimgrace@Kimgrace:~/lab4$ ./ls3 -t ls3_2
ls3_2:
drwxr-xr-z 2kimgracekimgrace 4096 Oct  4 22:03 .
-rw-r--r-- 1kimgracekimgrace 3050 Oct  4 22:05 ls3.c
drwxrwxr-z 3kimgracekimgrace 4096 Oct 26 20:03 ..
전체 파일 개수: 3
전체 파일 크기: 11242

```

▼ 과제 3

1. my_cp src_file dest_file
 - a. src_file 을 dest_file 에 복사하는 프로그램
 - b. open/read/write/close 시스템 콜을 사용한다.
 - 다음은 src_file의 코드이다. fs를 사용해서, data2.bin 파일을 읽어온다. read, write 가 가능하며, close(fs)를 사용해 종료한다.

- 작성한 후, 파일을 dest_file로 복사해야 하는데, 명령어로 간단하게 할 수가 있다.—
> cp src_file.c dest_file.c

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
    int fs;
    int a = 10;
    int b = 20;
    fs = open("data2.bin", O_RDWR|O_CREAT);
    read(fs, &a, sizeof(int));
    read(fs, &b, sizeof(int));
    write(fs, &a, sizeof(int));
    write(fs, &b, sizeof(int));
    close(fs);
}
```

- 데이터를 3개 정도 가지는 구조체를 하나 선언하고, 그 데이터를 여러 개(예: 5개)를 파일에 쓰는 프로그램과 반대로 그 파일에서 본인이 write한 구조체 데이터를 읽어서 출력하는 프로그램 작성.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

struct Node
{
    int a;
    int b;
    char name[200];
};

int main()
{
    int fb;
    struct Node node;
    node.a = 10;
    node.b = 20;
    strcpy(node.name, "kim");
    fb = open("data3.bin", O_RDWR|O_CREAT);
    write(fb, &node, sizeof(struct Node));
    read(fb, &node, sizeof(struct Node));
    printf("node.a = %d node.b = %d node.name = %s \n", node.a, node.b, node.name);
    close(fb);
    fb = open("data3.bin", O_RDWR|O_CREAT);
    write(fb, &node, sizeof(struct Node));
    read(fb, &node, sizeof(struct Node));
    printf("node.a = %d node.b = %d node.name = %s \n", node.a, node.b, node.name);
    close(fb);
    fb = open("data3.bin", O_RDWR|O_CREAT);
    write(fb, &node, sizeof(struct Node));
    read(fb, &node, sizeof(struct Node));
```

```

    printf("node.a = %d node.b = %d node.name = %s \n", node.a, node.b, node.name);
    close(fb);
    fb = open("data3.bin", O_RDWR|O_CREAT);
    write(fb, &node, sizeof(struct Node));
    read(fb, &node, sizeof(struct Node));
    printf("node.a = %d node.b = %d node.name = %s \n", node.a, node.b, node.name);
    close(fb);
    fb = open("data3.bin", O_RDWR|O_CREAT);
    write(fb, &node, sizeof(struct Node));
    read(fb, &node, sizeof(struct Node));
    printf("node.a = %d node.b = %d node.name = %s \n", node.a, node.b, node.name);
    close(fb);
}

```

▼ 과제 4

1. lab6 디렉토리 내부에 여러 디렉토리를 생성하고 특정 디렉토리에 여러 개의 .c OR .txt 파일에 나의 이름을 넣어둔다. 그 후, lab6 디렉토리에서 다음 두 명령어를 실행한다.

- a. find . -name “*.c” -print
- b. find . -name “*.c” -exec grep (내 이름) {} /dev/null \;
- a, b, c 라는 각각의 디렉토리를 생성해 준다.
 - lab6 — a — a.c b.c
 - b — x.c
 - c — d1(dir) — w.c
 - d2(dir) — v.c

이렇게 작성 해 준 후 , 위에서 제시한 명령어를 실행하면, 내가 찾고자 하는 것이 어떤 C 파일에 있는지 출력해 준다.

2. sigdemo.c 파일을 작성해서 실행을 해 보는데 이 때, 나의 이름이 출력되도록 해 본다. sigdemo1을 실행하고 있을 때, 다른 창에서 ps -aux 명령어로 해당 프로세서의 id를 찾고 , kill -2 프로세스ID 를 실행해서 나의 이름이 잘 출력되는지 확인한다.

Hello	kimgrace	488	0.0	0.0	2492	516	pts/1	T
Hello	root	527	0.0	0.0	2184	368	?	Ss
kimgrace	root	528	0.0	0.0	2184	380	?	S
Hello	kimgrace	529	0.0	0.0	10056	5148	pts/2	Ss
Hello	kimgrace	545	0.0	0.0	2492	576	pts/1	T
Hello	kimgrace	573	0.0	0.0	2492	580	pts/2	S+
Hello	kimgrace	574	0.0	0.0	10860	3404	pts/1	R+
Hello	kimgrace@Kimgrace:~/lab6\$	kill	-2	574				
Hello	bash:	kill:	(574)	-	No such process			
Hello	kimgrace@Kimgrace:~/lab6\$	kill	-2	573				
kimgrace	kimgrace@Kimgrace:~/lab6\$	kill	-2	573				
Hello	kimgrace@Kimgrace:~/lab6\$	kill	-2	573				
Hello	kimgrace@Kimgrace:~/lab6\$	kill	-2	573				
Hello	kimgrace@Kimgrace:~/lab6\$							

잘 나오는 것을 확인 할 수 있다. 궁금한게 저기 ps -aux 명령어 쳤을 때, 저 당시 기준 오후 8시 58분 경인데, 이 시간에 해당하는 ID가 두 개가 뜨더라, 그래서 맨 밑에 있는 574로 했는데 No such process 가 떠서 574로 하니깐 실행이 정상적으로 되었다.

왜 그럴까 하고 생각을 해 보았다. 다른 건 뭐 딱히 큰 차이가 없긴 한데, 가장 큰 차이가 COMMAND에서 ./sigdemo, ps -aux 가 찍혀있다. 아마, ps -aux 명령어가 들어간 ID로 실행을 해야 해서 다른 COMMAND 즉, ./sigdemo가 해당하는 ID를 입력하니깐 오류가 나서 저렇게 뜬 것 같다는 추측이 든다.

▼ 과제 5

- 5초에 한번씩 자신의 이름과 현재 시각을 출력한다. 이를 3회 출력이 되게끔 한 다음, 프로그램을 종료한다. (alarm(), SIGALRM 핸들러 사용)
 - b.c 파일에 작성했고, 실행 파일 이름을 b.out으로 했다.

```
#include <stdio.h>
#include <time.h>
#include <signal.h>

void my_alarm_handler(int signum)
{
    printf("Kimgrace\n");
    alarm(5);
}
int main()
{
    time_t ct;
    struct tm tm;
    ct = time(NULL);
    tm = *localtime(&ct);
    signal(SIGALRM, my_alarm_handler);
    alarm(5);
    for (int i = 0; i < 3; i++)
    {
        printf("hour: %d, min : %d, sec : %d\n", tm.tm_hour, tm.tm_min, tm.tm_sec);
        sleep(5);
    }
}
```

```
kimgrace@Kimgrace:~/lab7$ ./b.out
hour: 21, min : 14, sec : 4
Kimgrace
hour: 21, min : 14, sec : 4
Kimgrace
hour: 21, min : 14, sec : 4
Kimgrace
```

2. usr1.c 와 usr2.c 를 프로그램을 작성해서 다음의 기능을 구현한다.

- a. 각각 자신의 pid 출력
 - b. 각각 상대의 pid 입력
 - c. usr1 → usr2 에서 SIGUSR2전송
 - d. usr2 는 SIGUSR2를 받으면, usr1 에게 SIGUSR1 전송
 - e. 그 이후에 각각 프로그램 종료한다.
- 그럼, usr1.c && usr2.c 코드를 작성한다.

```
//usr1.c
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>

int usr2_pid;
void usr1_sig_handler(int signum)
{
    printf("I'm usr1\n");
    alarm(2);
}
void main()
{
    pid_t pid;
    pid = getpid();
    printf("usr1 pid = %d\n", pid);
    printf("You are usr1, plz Enter the usr2_pid\n");
    scanf("%d", &usr2_pid);
    signal(SIGUSR1, usr1_sig_handler);
    kill(usr2_pid, SIGUSR2);
    pause();
}

//usr2.c
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>

int usr1_pid;
void usr2_sig_handler(int signum)
{
    printf("I'm usr2\n");
    kill(usr1_pid, SIGUSR1);
    exit(0);
}
void main()
{
    pid_t pid;
    pid = getpid();
    printf("usr2 pid = %d\n", pid);
    printf("You are usr2, plz Enter the usr1 pid\n");
    scanf("%d", &usr1_pid);
    signal(SIGUSR2, usr2_sig_handler);

    while(1)
```

```

    {
        printf("Hello, usr2\n");
        sleep(2);
    }
}

```

usr1.c 와 usr2.c 는 코드를 자세히 보면, 다르다. 필자는 처음에는 서로의 코드를 똑같이 작성했는데, 반복문이 실행이 되지 않아서 조교님께 조언을 구했다.

우선 유저2는 유저1의 ID 시그널을 받으면, 종료하고 반복문을 탈출하도록 해야 한다. exit(0)을 안 쓰면 시그널 신호를 받은 것만 뜨고 반복은 멈추지 않는다.

유저1의 신호를 받는 코드는 kill(usr1_pid, SIGUSR1); 이렇게 작성한다.

일단 pid_t 선언해 주고 유저1 아이디 받는 것 까지는 그냥 작성하면 된다.

```

pid_t pid;
pid = getpid();
printf("usr2 pid = %d\n", pid);
printf("You are usr2, plz Enter the usr1 pid\n");
scanf("%d", &usr1_pid);

while(1)
{
    printf("Hello, usr2\n");
    sleep(2);
} // 그리고 반복문 코드

```

이제 유저1의 코드를 분석해 보면,

유저1은 신호를 줘야하기 때문에 alarm을 쓴다. **alarm 은 신호를 주는 것이고, kill 은 신호를 받는 것이다.**

그리고 main()함수로 돌아와서 pid_t 선언하고 유저2의 신호 입력받는 것 까지는 똑같다. 그리고 유저1의 신호를 받는 것 또한 동일하다.

```
signal(SIGUSR1, usr1_sig_handler);
```

그리고 유저2의 신호를 받으면, 프로세서가 I'm usr1을 출력하고 정지한다.

```
kill(usr2_pid, SIGUSR2);
pause();
```

```

$ gcc -o killtest timetest usr1.c usr2.out
$ ./killtest
$ ./killtest.c usr1 usr2.c
<imgrace@Kimgrace:~/lab7$ ./usr1.out
usr1 pid = 596
You are usr1, plz Enter the usr2_pid
598
I'm usr1
<imgrace@Kimgrace:~/lab7$ vi usr1.c
<imgrace@Kimgrace:~/lab7$ vi usr2.c
<imgrace@Kimgrace:~/lab7$ vi usr1.c
Kimgrace@Kimgrace:~/lab7$ ./usr2.out
usr2 pid = 598
You are usr2, plz Enter the usr1 pid
596
Hello, usr2
I'm usr2

```

(실행 창 2개로 함)

▼ 과제 6

- psh1.c 를 수정하여 계속해서 여러 명령어를 수행하는 프로그램을 완성하시오.
- 매번 본인의 이름이 출력이 되도록 수정함 (ex: Kyong Arg[%d]?)

```
//Ppt 자료 psh1.c 코드 수정 (교수님 ver.)  
  
#include <stdio.h>  
#include <unistd.h>  
#include <string.h>  
  
int main ()  
{  
    char *arglist[10];  
    char buf[100];  
    int i;  
    i = 0;  
    while (i < 10) {  
        printf ("Arg[%d]? ", i);  
        gets (buf);  
        arglist[i] = (char*) malloc (strlen(buf) + 1);  
        strcpy (arglist[i], buf);  
        if (strcmp (arglist[i], "") == 0) {  
            arglist[i] = NULL;  
            break;  
        }  
        i++;  
    }  
    execvp (arglist[0], arglist);  
}
```

```

kimgrace@Kimgrace:~/lab8$ ./edit_psh1
Arg[0]? ls
Arg[1]? -al
Arg[2]?
total 272
drwxr-xr-x  2 kimgrace kimgrace  4096 Nov  6 20:40 .
drwxr-xr-x 19 kimgrace kimgrace  4096 Nov  6 20:40 ..
-rw-r--r--  1 kimgrace kimgrace   274 Nov  3 09:46 a.c
-rw xr-xr-x  1 kimgrace kimgrace 16880 Nov  3 09:46 a.out
-rw xr-xr-x  1 kimgrace kimgrace 16744 Nov  4 18:12 after
-rw-r--r--  1 kimgrace kimgrace  149 Nov  4 18:14 after.c
-rw-r--r--  1 kimgrace kimgrace  153 Nov  3 09:42 b.c
-rw xr-xr-x  1 kimgrace kimgrace 16824 Nov  3 09:43 b.out
-rw xr-xr-x  1 kimgrace kimgrace 16840 Nov  4 18:13 before
-rw-r--r--  1 kimgrace kimgrace  221 Nov  4 18:16 before.c
-rw xr-xr-x  1 kimgrace kimgrace 16976 Nov  6 20:40 edit_psh1
-rw-r--r--  1 kimgrace kimgrace  379 Nov  6 20:40 edit_psh1.c
-rw xr-xr-x  1 kimgrace kimgrace 16792 Nov  4 17:54 exec1
-rw-r--r--  1 kimgrace kimgrace  258 Nov  6 20:39 exec1.c
-rw xr-xr-x  1 kimgrace kimgrace 16952 Nov  3 21:22 forkdemo1
-rw-r--r--  1 kimgrace kimgrace  451 Nov  4 11:53 forkdemo1.c
-rw xr-xr-x  1 kimgrace kimgrace 16832 Nov  4 18:28 forkdemo2
-rw-r--r--  1 kimgrace kimgrace  299 Nov  4 18:28 forkdemo2.c
-rw xr-xr-x  1 kimgrace kimgrace 16792 Nov  4 18:32 forkdemo3
-rw-r--r--  1 kimgrace kimgrace  177 Nov  4 18:30 forkdemo3.c
-rw xr-xr-x  1 kimgrace kimgrace 16832 Nov  4 18:34 forkdemo4
-rw-r--r--  1 kimgrace kimgrace  330 Nov  4 18:33 forkdemo4.c
-rw xr-xr-x  1 kimgrace kimgrace 17128 Nov  3 11:30 psh1
-rw-r--r--  1 kimgrace kimgrace  548 Nov  6 20:38 psh1.c

```

```

#include <stdio.h>
#include <unistd.h>
#include <string.h>

int child_task()
{
    char *arglist[10];
    char buf[100];
    int i;
    i = 0;
    while(i < 10){
        printf("Kimgrace Arg[%d]?", i);
        gets(buf);
        arglist[i] = (char*)malloc(strlen(buf)+1);
        strcpy(arglist[i], buf);
        if(strcmp(arglist[i], "") == 0) {
            arglist[i] = NULL;
            break;
        }
        i++;
    }
    execvp(arglist[0], arglist);
}

int main(){
    pid_t childId = 0;

```

```

while(1){
    childId = fork();
    switch(childId){
        case 0:
            child_task();
        default:
            wait();
            printf("child done\n");
    }
}
}

```

```

kimgrace@Kimgrace:~/lab8$ ./psh1
Kimgrace Arg[0]?ls
Kimgrace Arg[1]?-al
Kimgrace Arg[2]?
total 248
drwxr-xr-x  2 kimgrace kimgrace  4096 Nov  6 20:38 .
drwxr-xr-x 19 kimgrace kimgrace  4096 Nov  6 20:38 ..
-rw-r--r--  1 kimgrace kimgrace   274 Nov  3 09:46 a.c
-rw xr-xr-x  1 kimgrace kimgrace 16880 Nov  3 09:46 a.out
-rw xr-xr-x  1 kimgrace kimgrace 16744 Nov  4 18:12 after
-rw-r--r--  1 kimgrace kimgrace   149 Nov  4 18:14 after.c
-rw-r--r--  1 kimgrace kimgrace   153 Nov  3 09:42 b.c
-rw xr-xr-x  1 kimgrace kimgrace 16824 Nov  3 09:43 b.out
-rw xr-xr-x  1 kimgrace kimgrace 16840 Nov  4 18:13 before
-rw-r--r--  1 kimgrace kimgrace   221 Nov  4 18:16 before.c
-rw xr-xr-x  1 kimgrace kimgrace 16792 Nov  4 17:54 exec1
-rw-r--r--  1 kimgrace kimgrace   258 Nov  4 18:01 exec1.c
-rw xr-xr-x  1 kimgrace kimgrace 16952 Nov  3 21:22 forkdemo1
-rw-r--r--  1 kimgrace kimgrace   451 Nov  4 11:53 forkdemo1.c
-rw xr-xr-x  1 kimgrace kimgrace 16832 Nov  4 18:28 forkdemo2
-rw-r--r--  1 kimgrace kimgrace   299 Nov  4 18:28 forkdemo2.c
-rw xr-xr-x  1 kimgrace kimgrace 16792 Nov  4 18:32 forkdemo3
-rw-r--r--  1 kimgrace kimgrace   177 Nov  4 18:30 forkdemo3.c
-rw xr-xr-x  1 kimgrace kimgrace 16832 Nov  4 18:34 forkdemo4
-rw-r--r--  1 kimgrace kimgrace   330 Nov  4 18:33 forkdemo4.c
-rw xr-xr-x  1 kimgrace kimgrace 17128 Nov  3 11:30 psh1
-rw-r--r--  1 kimgrace kimgrace   548 Nov  6 20:38 psh1.c
child done
Kimgrace Arg[0]?

```

▼ 과제 7

- wtest 에 구현 할 것 => ProgramA
 1. 키보드로 입력받을 문자열을 rtest에 전달
 2. rtest에서 msg + "name" 으로 출력된 문자열을 받아오고 이를 다시 출력한다.
이 과정을 end로 입력받을 때까지 반복한다.
- rtest 에 구현 할 것 => ProgramB
 1. wtest로부터 문자열 입력 대기 및 받은 문자열 출력
 2. 새로운 문자열 생성 : strcat -> msg + "name" & 생성한 문자열을 다시 wtest에 전달 한다.
A가 end 를 입력할 때까지 반복

```

//programA.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>

int main()
{
char msg[100];
int nread;
int fd2;
int end = 0;
int fd1;
char msg2[100];

```
fd1 = open("f1", O_RDONLY);
fd2 = open("f2", O_RDWR);```
while(1)
{
 printf("Enter a string:\\n");
 gets(msg);
 nread = write(fd2, msg, strlen(msg)+1);
 printf("Reader: %s\\n", msg);

 nread = read(fd1, msg2, sizeof(msg2));
 msg2[nread] = 0;
 printf("response the programB: %s\\n", msg2);

 if(strncmp("end", msg, 3) == 0) {
 break;
 }
}
}

```

```

//programB.c
#include <stdio.h>
#include <fcntl.h>
#include <string.h>

int main()
{
char msg[100];
int fd1, fd2;
int nread;
int end = 0;
char msg2[100];

fd1 = open("f1", O_RDWR);
fd2 = open("f2", O_RDONLY);
while(1){
 nread = read(fd2, msg, sizeof(msg));
 msg[nread] = 0;
 printf("Reader : %s\\n", msg);
 strcpy(msg2, msg);
}

```

```

 strcat(msg2, " kimgrace");
 write(fd1, msg2, strlen(msg2));
 printf("New string : %s\n", msg2);

 if(end == 1)
 {
 break;
 }
 if (strcmp("end", msg) == 0)
 {
 break;
 }
 }
}

```

```

kimgrace@Kingrace:~$ cd lab9
kimgrace@Kingrace:~/lab9$ ls
a.txt c.out f1 ftest pipetest.c programA.c ptest test.txt
b.txt f1 ftest pipetest.c programB.c readtest.c writetest.c
c.c f2 ftest.c programA.c programB.c rtest wtest
kimgrace@Kingrace:~/lab9$./programA
Enter a string:
Hello
Reader: Hello
response the programB: Hello kimgrace
Enter a string:
Hi
Reader: Hi
response the programB: Hi kimgrace
Enter a string:
end
Reader: end
response the programB: end kimgrace
kimgrace@Kingrace:~/lab9$

```

```

b.txt f1 ftest pipetest.c programB.c readtest.c writetest.c
c.c f2 ftest.c programA.c programB.c rtest wtest
kimgrace@Kingrace:~/lab9$./programB
Reader : Hello
New string : Hello kimgrace
Reader : Hi
New string : Hi kimgrace
Reader : end
New string : end kimgrace
kimgrace@Kingrace:~/lab9$

```

## ▼ 과제 8

Message를 사용해서 과제 7 내용 수행하기

```

//programA
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/types.h>

int main()
{
 int msgqid;
 key_t key;
 char buf[100];
 key = 123;
 int end = 0;

 while(1)
 {
 msgqid = msgget(key, IPC_CREAT|0666);
 printf("Enter a string to send:");
 gets(buf);
 msgsnd(msgqid, buf, strlen(buf)+1, 0);

 msgrcv(msgqid, buf, 100, 0, 0);
 printf("buf = [%s]\n", buf);
 }
}

```

```

 if (strcmp("end", buf, 3) == 0){
 break;
 }
 }

//programB
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <sys/types.h>
#include <fcntl.h>
#include <string.h>

int main()
{
 int msgqid;
 key_t key; //각 프로세서의 고유 이름
 char buf[100];
 key = 123;
 char buf2[100];
 int end = 0;
 msgqid = msgget(key, IPC_CREAT | 0666);

 while(1)
 {
 msgrcv(msgqid, buf, 100, 0, 0);
 printf("buf = [%s]\n", buf);
 strcpy(buf2, buf);
 strcat(buf2, "Kimgrace");
 printf("New string: %s\n", buf2);

 printf("Enter a string to send:");
 gets(buf);
 msgsnd(msgqid, buf, strlen(buf)+1, 0);

 if(end == 1)
 {break;}
 if(strcmp("end", buf) == 0)
 {
 break;
 }
 }
}

```

```

C:\WINDOWS\system32\cm * kimgrace@Kimgrace:~/lab10 * + - x
kimgrace@Kimgrace:~$ cd lab10
kimgrace@Kimgrace:~/lab10$./programA
Enter a string to send:Hello
buf = [Hi]
Enter a string to send:Hello world
buf = [end]
kimgrace@Kimgrace:~/lab10$ |

C:\WINDOWS\system32\cm * kimgrace@Kimgrace:~/lab10 * + - x
kimgrace@Kimgrace:~$ cd lab10
kimgrace@Kimgrace:~/lab10$./programB
buf = [Hello]
New string: HelloKimgrace
Enter a string to send:Hi
buf = [Hello world]
New string: Hello worldKimgrace
Enter a string to send:end
kimgrace@Kimgrace:~/lab10$ |

```

## ▼ 과제 9

Shared memory(semaphore를 이용해서 과제 7과 동일하게 내용 주고받고, 대기하고 , end 입력시 종료하는 프로그램 작성하기)

### (1) Shared Memory 이용한 메시지 전달하기 (동기화를 위해서 Semaphore 2개 사용 필요)

- program A

- + 키보드로 입력받은 문자열을 공유메모릴에 write한다.
- + 프로그램 B가 공유 메모리에 새로운 문자열을 쓰기를 기다린다.
- + 그리고, 그 문자열을 출력한다.
- + 이 과정을 "end"를 입력할 때까지 반복한다.

- program B

- + program A가 새로운 문자열을 write하기를 기다린다.
- + 그 문자열을 출력하고, 새로운 문자열을 만든다. ("예: strcat을 사용해서 msg + "이름")
- + 새로 만든 문자열을 공유메모리에 write한다.
- + 이 과정을 program A가 "end"를 받을 때까지 반복한다.

```
//programA.c
#include <fcntl.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/types.h>

#define BUF_SIZE 100
int main() {
 int shmid;
 sem_t *sndsem, *recvsem;
 char *sndbuf, *recvbuf;

 // initialize semaphore
 // sem_1: programA -> programB
 sndsem = sem_open("testsem-1", O_CREAT, 0666, 0);
 // sem_2: programB -> programA
 recvsem = sem_open("testsem-2", O_CREAT, 0666, 0);

 // initialize shared memory
 shmid = shmget(1234, BUF_SIZE, IPC_CREAT | 0666);
 sndbuf = shmat(shmid, NULL, 0);

 shmid = shmget(1235, BUF_SIZE, IPC_CREAT | 0666);
 recvbuf = shmat(shmid, NULL, 0);

 int end = 0;

 while (1) {
 printf("Enter a string to send:");
 gets(sndbuf);

 // write message to programB via shared memory
 printf("sndbuf = [%s]\n", sndbuf);
```

```

 sem_post(sndsem);

 // wait for message from programB
 sem_wait(recvsem);
 printf("recvbuf = [%s]\n", recvbuf);

 if (strncmp("end", sndbuf, 3) == 0) {
 break;
 }
}
}

```

```

//programB.c
#include <fcntl.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/IPC.h>
#include <sys/shm.h>
#include <sys/types.h>

#define BUF_SIZE 100
int main() {
 int shmid;
 sem_t *sndsem, *recvsem;
 char *sndbuf, *recvbuf;

 // initialize semaphore
 // sem_1: programB -> programA
 sndsem = sem_open("testsem-2", O_CREAT, 0666, 0);
 // sem_2: programA -> programB
 recvsem = sem_open("testsem-1", O_CREAT, 0666, 0);

 // initialize shared memory
 shmid = shmget(1235, BUF_SIZE, IPC_CREAT | 0666);
 sndbuf = shmat(shmid, NULL, 0);

 shmid = shmget(1234, BUF_SIZE, IPC_CREAT | 0666);
 recvbuf = shmat(shmid, NULL, 0);

 int end = 0;

 while (1) {
 // wait for message from programA
 sem_wait(recvsem);
 printf("recvbuf = [%s]\n", recvbuf);

 // write message to programB via shared memory
 strcpy(sndbuf, recvbuf);
 strncat(sndbuf, " postfix");
 sem_post(sndsem);
 printf("sndbuf = [%s]\n", sndbuf);

 if (strncmp("end", recvbuf, 3) == 0) {
 break;
 }
 }
}

```

The screenshot shows two terminal windows side-by-side. The left window shows the source code for a client program named 'programB.c' which reads strings from standard input and sends them to a server. The right window shows the source code for a server program named 'programA.c' which receives strings from a client and prints them to standard output. Both programs use socket programming to communicate.

```

C:\WINDOWS\system32\cmd.exe | kimgrace@Kingrace:~/lab11 |
kimgrace@Kingrace:~/lab11$ vi programB.c
kimgrace@Kingrace:~/lab11$./programB
recvbuf = [Helloo]
sndbuf = [Helloo Kim_grace]
recvbuf = [Hello world]
sndbuf = [Hello world Kim_grace]
recvbuf = [Hi]
sndbuf = [Hi Kim_grace]
recvbuf = [end]
sndbuf = [end Kim_grace]
kimgrace@Kingrace:~/lab11$ |

C:\WINDOWS\system32\cmd.exe | kimgrace@Kingrace:~/lab11 |
kimgrace@Kingrace:~/lab11$./programA
Enter a string to send:Helloo
recvbuf = [Helloo]
sndbuf = [Helloo Kim_grace]
Enter a string to send:Hello world
recvbuf = [Hello world]
sndbuf = [Hello world Kim_grace]
Enter a string to send:Hi
recvbuf = [Hi]
sndbuf = [Hi Kim_grace]
Enter a string to send:end
recvbuf = [end]
sndbuf = [end Kim_grace]
kimgrace@Kingrace:~/lab11$

```

## ▼ 과제 10

Thread를 사용해서 코드를 작성하는데, 2개 이상의 thread를 사용해서 코딩을 한다. 이 때, 숫자가 커지면 thread1이 먼저 끝나고 그 한참 후에, thread2가 끝나는데, 이걸 차이가 거의 없게, 동시에 끝나도록 할 수 있는 코드를 작성하여라.

(1) 2부터 N까지의 소수(Prime Number)의 개수를 구하는 프로그램을 Thread로 작성하시오.

단, 여러 개의 Thread를 생성하더라도 항상 Thread들이 거의 동시에 끝나도록 한다.

\* 알고리즘 예

- 전역변수 : int total\_cnt; int testNum;
- 전역 mutex 변수 : pthread\_mutex\_t total\_cnt\_lock; pthread\_mutex\_t testNum\_lock;
- Thread 주요 알고리즘

```

while (testNum <= N) {
 testNum에 대한 lock 걸기
 num <- testNum
 testNum 1 증가
 testNum에 대한 lock 풀기
 num이 소수이면 total_count 1증가
}

```

- 이번주 과제 내용 요약 : 사용자가 어떠한 일을 던져준다.(N개 만큼), 그러면, Key라는 것을 전역변수로 선언해 주는데, (total\_cnt, testNum)이 두개는 각각의 프로세서를 열람하고 수행할 수 있는 역할을 해 준다. 그럼 얘네를 열어주고 닫아주는 데에는 사용자가 생성한 만큼의 Key, 그러니까 해당 프로세서에 대한 전용 키이니 열었으면,(pthread\_mutex\_t total\_cnt\_lock, pthread\_mutex\_t testNum\_lock), 그 후, 해당 프로세스를 전용 키로 닫아줘야 하므로 pthread\_mutex\_lock(&{key});한다.

- Key create : **pthread\_mutex\_t** total\_cnt
- Key lock : **pthread\_mutex\_lock(&total\_cnt)**
- Key unlock : **pthread\_mutex\_unlock(&total\_cnt)** //이거 안 해주면 무한 lock에 걸려 버림
- 위에 3개는 암기하고 늘 세트로 외워두고 사용 할 것!

```
#include <stdio.h>
#include <pthread.h>
#include <time.h>

int isPrime (int n)
{
 for (int i = 2; i <= n/2; i++)
 if (n % i == 0)
 return 0;

 return 1;
}

int N;
int total_count = 0;//일의 결과를 담는 변수
//일을 나누는 변수
int testNum = 2; //0과 1은 검사 X
pthread_mutex_t count_lock;
pthread_mutex_t testNum_lock; //key create

void *prime_count (void *arg)
{
 int num;

 while(testNum <= N)
 {
 pthread_mutex_lock(&testNum_lock);
 num = testNum;
 testNum++;
 pthread_mutex_unlock(&testNum_lock);
 if (isPrime(num)){
 pthread_mutex_lock(&count_lock);
 total_count++;
 pthread_mutex_unlock(&count_lock);
 }
 }
}
int main ()
{
 int count;
 int status;

 pthread_t thrd1, thrd2;
 pthread_mutex_init(&count_lock, NULL);
 pthread_mutex_init(&testNum_lock, NULL);

 printf ("Enter a positive integer : ");
 scanf ("%d", &N);

 pthread_create (&thrd1, NULL, prime_count, NULL); //thread 생성
}
```

```
pthread_create (&thrd2, NULL, prime_count, NULL);

pthread_join (thrd1, &status);//thread 끌어줌
pthread_join (thrd2, &status);
printf ("The number of prime numbers is %d. \n", total_count);
}
```

```
kimgrace@Kimgrace:~/lab12$./Assignment
Enter a positive integer : 50
The number of prime numbers is 15.
kimgrace@Kimgrace:~/lab12$./Assignment
Enter a positive integer : 9
The number of prime numbers is 4.
kimgrace@Kimgrace:~/lab12$./Assignment
Enter a positive integer : 50
The number of prime numbers is 15.
kimgrace@Kimgrace:~/lab12$./Assignment
Enter a positive integer : 60
The number of prime numbers is 17.
```