

## 실습보고서 3

교과목명	시스템프로그래밍		분반	001
제출일자	2021-12-13		교수자명	장희숙
팀명	이름	학번	이름	학번
초코파이썬칩	김기태	20173152		
	안대현	20173217		

### ■ 실습 주제

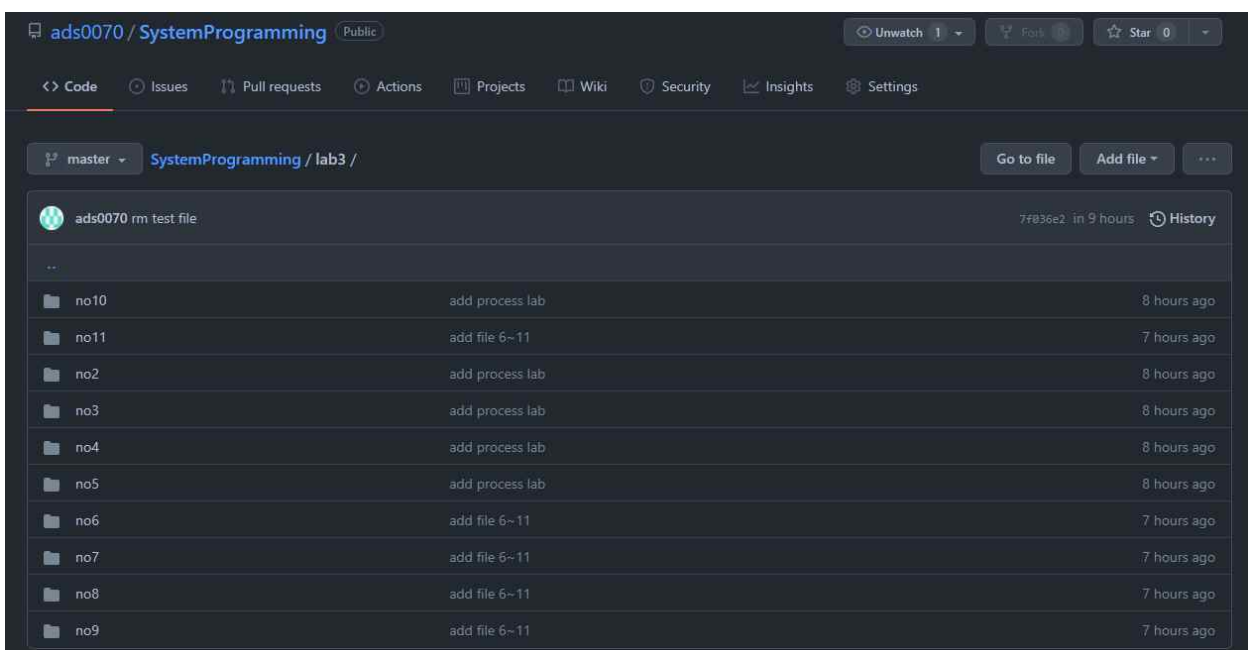
실습 3. 파일 처리, 프로세스 및 프로세스 간 통신

### ■ 실습 결과

#### 실습 순서

1. 자신의 github 저장소에 lab3 프로젝트를 생성하고 아래의 모든 과제 프로그램을 업로드한다.

git 주소 : <https://github.com/ads0070/SystemProgramming/tree/master/lab3>



## 2. 파일 및 디렉토리과 관련된 함수들을 사용하여 프로그램을 작성하고 실행하여보고, 익숙해지도록 사용해 본다.

### 1) 텍스트 파일 내용 복사 프로그램

```
file_copy.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <stdlib.h>

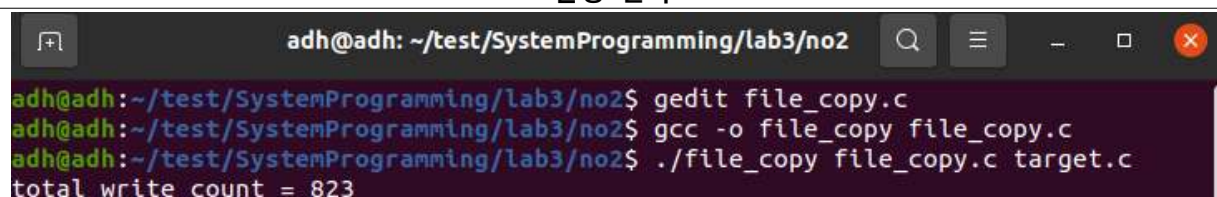
#define MAX_READ 10

int main(int argc, char *argv[])
{
    int src_fd;
    int dst_fd;
    char buf[MAX_READ];
    ssize_t rcnt;
    ssize_t tot_cnt = 0;

    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;

    if (argc < 3) {
        fprintf(stderr, "Usage: file_copy src_file dest_file\n");
        exit(1);
    }
    if ((src_fd = open(argv[1], O_RDONLY)) == -1) {
        perror("src open");
        exit(1);
    }
    if ((dst_fd = creat(argv[2], mode)) == -1) {
        perror("dst open");
        exit(1);
    }
    while ((rcnt = read(src_fd, buf, MAX_READ)) > 0) {
        tot_cnt += write(dst_fd, buf, rcnt);
    }
    if (rcnt < 0) {
        perror("read");
        exit(1);
    }
    printf("total write count = %ld\n", tot_cnt);
    close(src_fd);
    close(dst_fd);
}
```

### 실행 결과



```
adh@adh: ~/test/SystemProgramming/lab3/no2
adh@adh:~/test/SystemProgramming/lab3/no2$ gedit file_copy.c
adh@adh:~/test/SystemProgramming/lab3/no2$ gcc -o file_copy file_copy.c
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_copy file_copy.c target.c
total write count = 823
```

```
adh@adh:~/test/SystemProgramming/lab3/no2$ diff file_copy.c target.c
adh@adh:~/test/SystemProgramming/lab3/no2$
```

file\_copy.c의 실행 파일을 만들어 실행한다. 이때, 원본 파일을 file\_copy.c로 하고, 복사할 파일 이름을 target.c로 설정한다.

파일 내용이 제대로 복사되었는지 확인하기 위해 diff 명령을 사용하여 두 파일을 비교한다. 아무 것도 출력되지 않으면 두 파일이 동일하다는 뜻이다.

## 2) 파일 생성 프로그램

### file\_create.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

int main(int argc, char *argv[])
{
    int fd;
    char *buf = "This is a test.";
    ssize_t cnt;
    int flags = O_WRONLY | O_CREAT | O_TRUNC;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
    if (argc < 2) {
        fprintf(stderr, "Usage: file_create filename\n");
        exit(1);
    }
    if ( (fd = open(argv[1], flags, mode)) == -1 ){
        perror("open");
        exit(1);
    }
    cnt = write(fd, buf, strlen(buf));
    printf("write count = %ld\n", cnt);
    close(fd);
}
```

### 실행 결과

```
adh@adh: ~/test/SystemProgramming/lab3/no2
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_create t.txt
write count = 15
adh@adh:~/test/SystemProgramming/lab3/no2$ ls -al
합계 36
drwxrwxr-x 2 adh adh 4096 12월 12 22:10 .
drwxrwxr-x 3 adh adh 4096 12월 12 22:06 ..
-rwxrwxr-x 1 adh adh 17032 12월 12 22:06 file_create
-rw-rw-r-- 1 adh adh 602 12월 12 20:33 file_create.c
-rw-r--r-- 1 adh adh 15 12월 12 22:10 t.txt
adh@adh:~/test/SystemProgramming/lab3/no2$ cat t.txt
This is a test.adh@adh:~/test/SystemProgramming/lab3/no2$
```

file\_create.c의 실행 파일을 만들어 t.txt 파일을 생성하는 명령을 실행한다.

실행 결과로 write count = 15가 출력되고, cat 명령으로 t.txt 파일 내용을 출력하면 "This is a test." 문장이 나오는 것을 확인할 수 있다.

### 3) 파일 추가 모드 프로그램

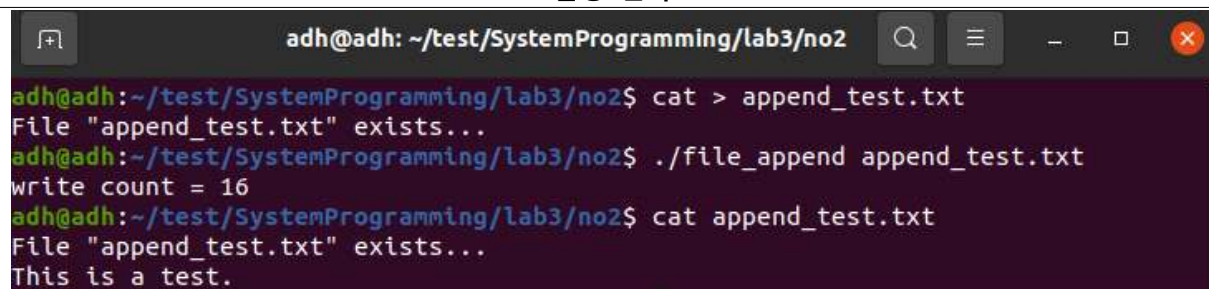
#### file\_append.c

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int fd;
    char *buf = "This is a test.\n";
    ssize_t cnt;
    int flags = O_WRONLY | O_CREAT | O_APPEND;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;

    if (argc < 2) {
        fprintf(stderr, "Usage: file_append filename\n");
        exit(1);
    }
    if ( (fd = open(argv[1], flags, mode)) == -1 ){
        perror("open");
        exit(1);
    }
    cnt = write(fd, buf, strlen(buf));
    printf("write count = %ld\n", cnt);
    close(fd);
}
```

#### 실행 결과



```
adh@adh: ~/test/SystemProgramming/lab3/no2
adh@adh:~/test/SystemProgramming/lab3/no2$ cat > append_test.txt
File "append_test.txt" exists...
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_append append_test.txt
write count = 16
adh@adh:~/test/SystemProgramming/lab3/no2$ cat append_test.txt
File "append_test.txt" exists...
This is a test.
```

테스트를 위한 append\_test.txt 파일을 만들고, "File "append\_test.txt" exists..." 문장을 입력한다. file\_append.c의 실행 파일을 만들고, append\_test.txt 파일을 매개변수로 입력하여 file\_append 파일을 실행한다.

실행 결과로 write count = 16이 출력되고, cat 명령으로 append\_test.txt 파일 내용을 출력하면 원래 있던 "File "append\_test.txt" exists..." 문장 아래에 "This is a test." 문장이 추가된 것을 확인할 수 있다.

### 4) 파일 크기 알아내기 프로그램

#### file\_size.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```

#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

int main(int argc, char *argv[])
{
    int fd;
    off_t size;

    if (argc < 2) {
        fprintf(stderr, "Usage: file_size filename\n");
        exit(1);
    }

    if ( (fd = open(argv[1], O_RDONLY)) == -1 ){
        perror("open"); /* errno에 대응하는 메시지 출력됨 */
        exit(1);
    }
    size = lseek(fd, 0, SEEK_END);
    if (size < 0) {
        perror("lseek");
        exit(1);
    }
    printf("%s's size = %ld\n", argv[1], size);
    close(fd);
}

```

#### 실행 결과

```

adh@adh:~/test/SystemProgramming/lab3/no2$ gedit file_size.c
adh@adh:~/test/SystemProgramming/lab3/no2$ gcc -o file_size file_size.c
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_size file_size.c
file_size.c's size = 553
adh@adh:~/test/SystemProgramming/lab3/no2$ ls -al file_size.c
-rw-rw-r-- 1 adh adh 553 12월 12 22:22 file_size.c

```

file\_size.c의 실행 파일 file\_size를 만든다. 매개 변수로 file\_size.c를 입력하여 file\_size를 실행한다. 실행 결과로 file\_size.c 파일의 크기인 553이 출력되는 것을 확인할 수 있다. 해당 값이 정확한지 확인하기 위해 ls -al 명령으로 file\_size.c 파일의 크기를 출력하면 똑같이 553이 출력되는 것을 확인할 수 있다.

#### 5) 표준 입출력 예제 프로그램

##### file\_io.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define BUFSIZE 256

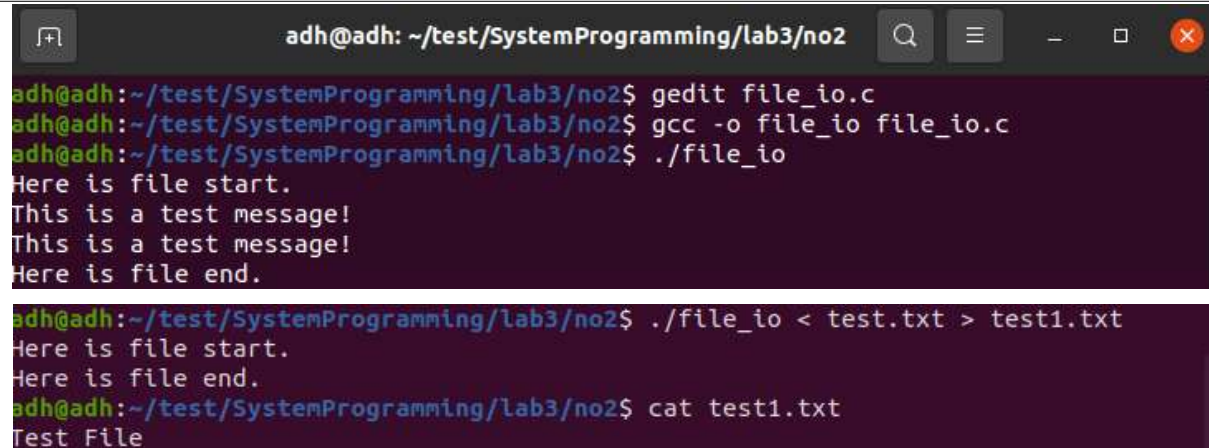
int main()
{
    int n;
    char buf[BUFSIZE];
    fprintf(stderr, "Here is file start.\n");

    while ((n = read(0, buf, BUFSIZE)) > 0)
        write (1, buf, n);
    fprintf(stderr, "Here is file end.\n");
}

```

```
}
    exit(0);
}
```

#### 실행 결과



```
adh@adh: ~/test/SystemProgramming/lab3/no2
adh@adh:~/test/SystemProgramming/lab3/no2$ gedit file_io.c
adh@adh:~/test/SystemProgramming/lab3/no2$ gcc -o file_io file_io.c
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_io
Here is file start.
This is a test message!
This is a test message!
Here is file end.

adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_io < test.txt > test1.txt
Here is file start.
Here is file end.
adh@adh:~/test/SystemProgramming/lab3/no2$ cat test1.txt
Test File
```

file\_io를 실행하면 "Here is file start." 문장이 출력된다. 다음으로, "This is a test message!" 문장을 입력한 뒤 Ctrl-D를 입력하면 아래에 다시 "This is a test message!" 문장이 출력되고, "Here is file end."가 출력된다.

file\_io를 실행할 때, test.txt로부터 표준 입력을 받도록 재지향하고, 표준 출력을 test1.txt로 재지향하면 test1.txt를 출력했을 때, "Test File"이 출력되는 것을 확인할 수 있다.

#### 6) 표준 출력 재지향 프로그램

##### file\_output.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int fd;
    char *buf = "This is a test output file.\n";
    int flags = O_WRONLY | O_CREAT | O_TRUNC;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
    if (argc < 2) {
        fprintf(stderr, "Usage: file_output filename\n");
        exit(1);
    }
    if ( (fd = open(argv[1], flags, mode)) == -1 ) {
        perror("open");
        exit(1);
    }
    if ( dup2(fd, 1) == -1 ) {
        perror("dup2");
        exit(1);
    }
    if ( close(fd) == -1 ) {
        perror("close");
        exit(1);
    }
}
```

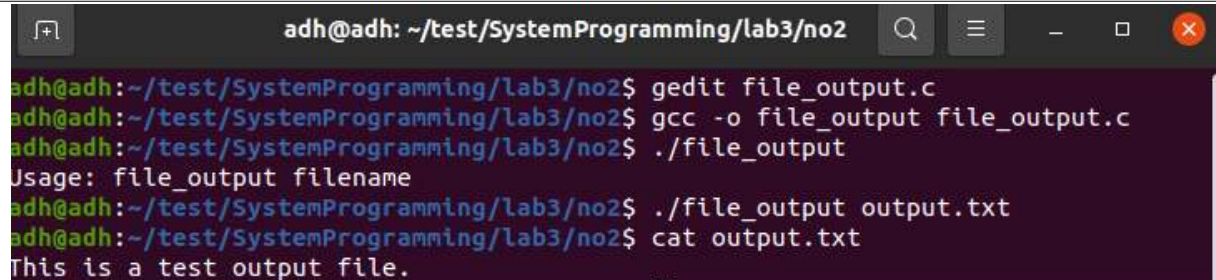


```

write(1, buf, strlen(buf));
exit(0);
}

```

#### 실행 결과



```

adh@adh: ~/test/SystemProgramming/lab3/no2
adh@adh:~/test/SystemProgramming/lab3/no2$ gedit file_output.c
adh@adh:~/test/SystemProgramming/lab3/no2$ gcc -o file_output file_output.c
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_output
Usage: file_output filename
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_output output.txt
adh@adh:~/test/SystemProgramming/lab3/no2$ cat output.txt
This is a test output file.

```

file\_output 실행파일을 만들어 실행할 때, 명령행 인수를 입력하지 않으면 “Usage: file\_output filename” 문장이 출력된다.

명령행 인수로 파일 이름 output.txt를 입력하여 실행하면 “This is a test output file” 문장의 표준 출력을 output.txt로 재지향하는 것을 확인할 수 있다.

### 7) 디렉토리 내용 보기 프로그램

#### file\_dir.c

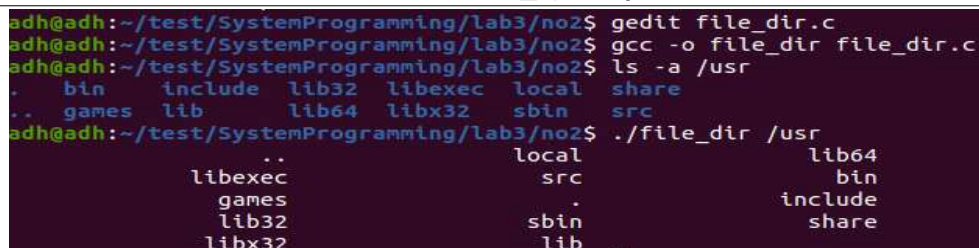
```

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>

int main(int argc, char *argv[])
{
    DIR *pdir;
    struct dirent *pde;
    int i = 0;
    if (argc < 2) {
        fprintf(stderr, "Usage: file_dir dirname\n");
        exit(1);
    }
    if ( (pdir = opendir(argv[1])) < 0 ) {
        perror("opendir");
        exit(1);
    }
    while ((pde = readdir(pdir)) != NULL) {
        printf("%20s ", pde->d_name);
        if (++i % 3 == 0)
            printf("\n");
    }
    printf("\n");
    closedir(pdir);
}

```

#### 실행 결과



```

adh@adh:~/test/SystemProgramming/lab3/no2$ gedit file_dir.c
adh@adh:~/test/SystemProgramming/lab3/no2$ gcc -o file_dir file_dir.c
adh@adh:~/test/SystemProgramming/lab3/no2$ ls -a /usr
.. bin include lib32 libexec local share
. games lib lib64 libx32 sbin src
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_dir /usr
.. local lib64
libexec bin
games . include
lib32 sbin share
libx32 lib

```

ls -a 명령으로 /usr 디렉토리 안에 들어있는 파일이나 폴더를 출력한다.

실행파일 file\_dir을 실행할 때, 명령행 인자로 /usr를 입력하여 실행한다. 실행 결과로, ls -a /usr의 결과와 같이 /usr 디렉토리 안에 있는 파일과 폴더 이름이 출력되는 것을 확인할 수 있다.

## 8) 디렉토리 이동 프로그램

### file\_chdir.c

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

#define MAX_BUF 256

int main(int argc, char *argv[])
{
    char buf[MAX_BUF];
    if (argc < 2) {
        fprintf(stderr, "Usage: file_chdir dirname\n");
        exit(1);
    }
    memset(buf, 0, MAX_BUF);

    if (getcwd(buf, MAX_BUF) < 0) {
        perror("getcwd");
        exit(1);
    }
    printf("working directory (before) = %s\n", buf);
    if (chdir(argv[1]) < 0) {
        perror("chdir");
        exit(1);
    }

    memset(buf, 0, MAX_BUF);
    if (getcwd(buf, MAX_BUF) < 0) {
        perror("getcwd");
        exit(1);
    }

    printf("working directory (after ) = %s\n", buf);
}
```

### 실행 결과

```
adh@adh:~/test/SystemProgramming/lab3/no2$ gedit file_chdir.c
adh@adh:~/test/SystemProgramming/lab3/no2$ gcc -o file_chdir file_chdir.c
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_chdir /usr
working directory (before) = /home/adh/test/SystemProgramming/lab3/no2
working directory (after ) = /usr
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_chdir ..
working directory (before) = /home/adh/test/SystemProgramming/lab3/no2
working directory (after ) = /home/adh/test/SystemProgramming/lab3
adh@adh:~/test/SystemProgramming/lab3/no2$ ./file_chdir /root
working directory (before) = /home/adh/test/SystemProgramming/lab3/no2
chdir: Permission denied
```

파일을 실행할 때, 명령행 인자로 경로를 설정하면 원래 있던 경로가 출력되고, 이동되는 경로가 출력된다. /root 파일로의 이동은 권한이 거부되었다고 출력된다.



3. 주어진 디렉토리 내에 존재하는 파일과 디렉토리를 나열하고, 디렉토리의 경우 재귀적으로 방문해서 그 디렉토리 내에 존재하는 파일과 디렉토리를 나열하는 프로그램을 작성하시오. (ls -R 명령과 동일한 결과를 보이도록)

list\_file\_dir.c

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>

#define DIR_MAX 20
#define FILE_MAX 100
int line_cnt = -1;

void visitDir(const char *name)
{
    DIR *dir;
    struct dirent *directory;

    char temp[DIR_MAX][FILE_MAX];
    char *directory_path[DIR_MAX];
    int index=0, p_index=0;

    for(int i =0; i < DIR_MAX; i++) {
        directory_path[i] =NULL;
        for(int j =0; j < FILE_MAX; j++) temp[i][j] = '\0';
    }

    if (!(dir = opendir(name)))
        return;

    printf("%s: \n", name);

    while ((directory = readdir(dir)) !=NULL) {
        if (directory -> d_type == DT_DIR) {
            char path[1024];
            if (strcmp(directory -> d_name, ".") ==0 || strcmp(directory -> d_name, "..") ==0
|| strcmp(directory -> d_name, ".git") ==0 )
                continue;

            snprintf(path, sizeof(path), "%s/%s", name, directory -> d_name);

            for(int k =0; k < strlen(path); k++)
                temp[index][k] = path[k];

            directory_path[index] = temp[index];
            index++;

            printf("%s\n", directory_path[index-1]);
        }
        else
            printf("%s\n", directory -> d_name);

        line_cnt++;
    }
}
```

```

        if(line_cnt > 5){
            printf("\n");
            line_cnt = 1;
        }
    }
    printf("\n\n");

    while(directory_path[p_index] != NULL) {
        if(directory_path[p_index] == NULL)
            break;
        visitDir(directory_path[p_index]);
        p_index++;
    }
    closedir(dir);
}

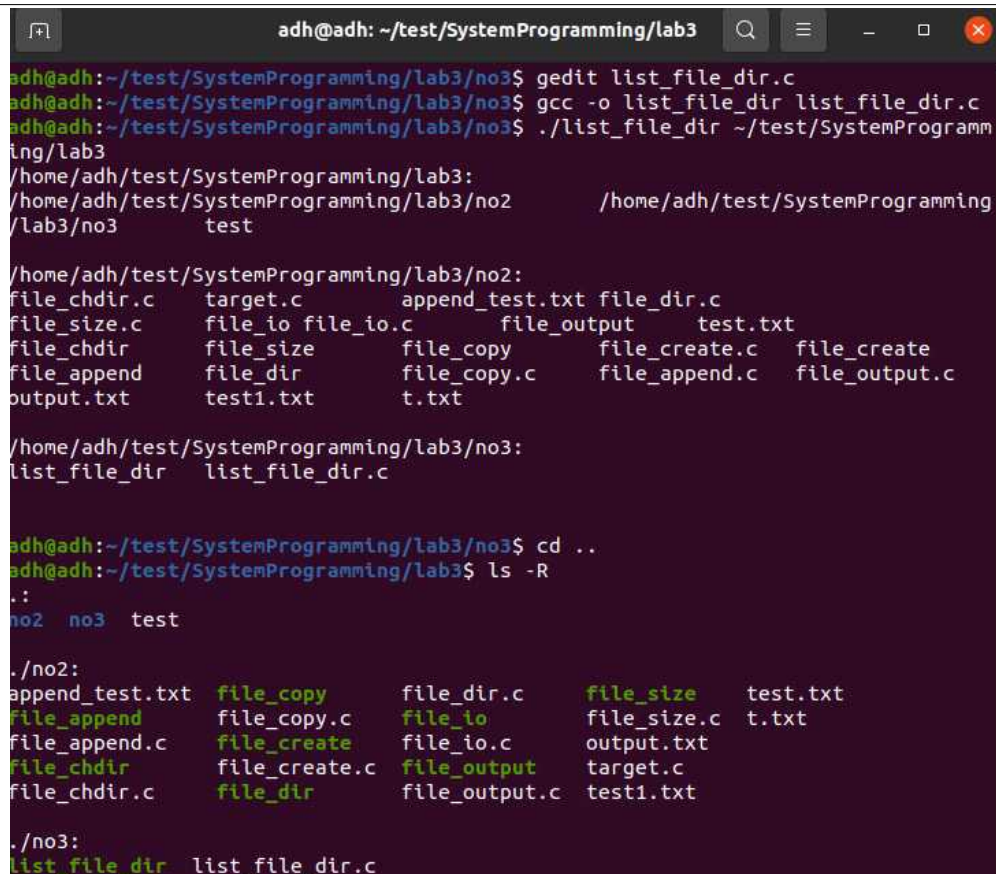
int main(int argc, char *argv[]) {
    char dir[1024];

    if(argc == 1)    strcpy(dir, ".");
    else strcpy(dir, argv[1]);

    visitDir(dir);
    return 0;
}

```

#### 실행 결과



```

adh@adh: ~/test/SystemProgramming/lab3
adh@adh:~/test/SystemProgramming/lab3/no3$ gedit list_file_dir.c
adh@adh:~/test/SystemProgramming/lab3/no3$ gcc -o list_file_dir list_file_dir.c
adh@adh:~/test/SystemProgramming/lab3/no3$ ./list_file_dir ~/test/SystemProgramm
ing/lab3
/home/adh/test/SystemProgramming/lab3:
/home/adh/test/SystemProgramming/lab3/no2      /home/adh/test/SystemProgramming
/lab3/no3      test

/home/adh/test/SystemProgramming/lab3/no2:
file_chdir.c  target.c      append_test.txt  file_dir.c
file_size.c   file_io file_io.c      file_output      test.txt
file_chdir    file_size   file_copy       file_create.c    file_create
file_append   file_dir    file_copy.c     file_append.c    file_output.c
output.txt    test1.txt   t.txt

/home/adh/test/SystemProgramming/lab3/no3:
list_file_dir  list_file_dir.c

adh@adh:~/test/SystemProgramming/lab3/no3$ cd ..
adh@adh:~/test/SystemProgramming/lab3$ ls -R
.:
no2  no3  test

./no2:
append_test.txt  file_copy      file_dir.c      file_size      test.txt
file_append      file_copy.c    file_io         file_size.c    t.txt
file_append.c    file_create    file_io.c       output.txt
file_chdir       file_create.c  file_output     target.c
file_chdir.c     file_dir       file_output.c   test1.txt

./no3:
list file dir  list file dir.c

```

파일을 실행할 때, 명령행 인자로 lab3 경로를 입력하여 lab3 디렉토리 내의 폴더나 파일을 탐색하도록 한다. 출력 결과로 lab3 내의 폴더 no2, no3와 파일 test가 출력되고, 디렉토리인 no2와 no3는 재귀 탐색을 하여 그 안에 포함된 파일들까지 출력한다.

ls -R 명령을 실행하면 출력 결과가 동일한 것을 확인할 수 있다.

#### 4. 몇 개의 문장을 타자시 잘못 타이핑한 횟수와 평균 분당 타자수를 측정하는 타자 연습 프로그램을 구현하시오.

```
typing.c

#include <stdio.h>
#include <stdlib.h>
#include <termios.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <math.h>

#define PASSWORDSIZE 12
typedef unsigned long long int tick64_t;
typedef unsigned long int tick32_t;

tick32_t GetTickCount()
{
    tick32_t tick = 0ul;
    #if defined(WIN32) || defined(WIN64)
        tick = GetTickCount();
    #else
        struct timespec ts;
        clock_gettime(CLOCK_MONOTONIC, &ts);
        tick = (ts.tv_sec*1000ul) + (ts.tv_nsec/1000ul/1000ul);
    #endif
    return tick;
}

int main(void) {
    int fd, start;
    int nread, cnt=0, cnt1=0, errcnt=0;
    double typing_msec, typing_sec, typing_per_sec;
    char ch, text[] = {"The magic thing is that you can change it."}, ch1, text1[] = {"Dokdo is Korean territory."};
    struct termios init_attr, new_attr;
    fd = open(ttname(fileno(stdin)), O_RDWR); tcgetattr(fd, &init_attr);
    new_attr = init_attr;
    new_attr.c_lflag &= ~ICANON;
    new_attr.c_lflag &= ~ECHO; /* ~(ECHO | ECHOE | ECHOK | ECHONL); */
    new_attr.c_cc[VMIN] = 1;
    new_attr.c_cc[VTIME] = 0;
    if (tcsetattr(fd, TCSANOW, &new_attr) != 0) {
        fprintf(stderr, "터미널 속성을 설정할 수 없음.\n");
    }
    printf("다음 문장을 그대로 입력하세요.\n%s\n", text);
    start = GetTickCount();
    while ((nread=read(fd, &ch, 1)) > 0 && ch != '\n') {
        if (ch == text[cnt++])
            write(fd, &ch, 1);
        else {
            write(fd, "*", 1); errcnt++;
        }
    }
    printf("\n다음 문장을 그대로 입력하세요.\n%s\n", text1);
    while ((nread=read(fd, &ch1, 1)) > 0 && ch1 != '\n') {
        if (ch1 == text1[cnt1++])
```

```

        write(fd, &ch1, 1);
    else {
        write(fd, "*", 1);
        errcnt++;
    }
}
typing_msec = GetTickCount() - start;
typing_sec = fmod((typing_msec/1000), 60);
typing_per_sec = (sizeof(text) / sizeof(int)) / typing_sec;
printf("\n걸린 시간: %.3f초\n", typing_sec);
printf("분당 %d타를 치셨습니다.\n\n", (int)(typing_per_sec*60));
printf("타이핑 오류의 횟수는 %d\n", errcnt);
tcsetattr(fd, TCSANOW, &init_attr);
close(fd);
}

```

#### 실행 결과

```

adh@adh:~/test/SystemProgramming/lab3/no4$ gcc typing.c -lm -o typing
adh@adh:~/test/SystemProgramming/lab3/no4$ ./typing
다음 문장을 그대로 입력하세요.
The magic thing is that you can change it.
The magi* thing is that you can change it.
다음 문장을 그대로 입력하세요.
Dokdo is Korean territory.
Dokdo is Korean territory.
걸린 시간: 19.305초
분당 31타를 치셨습니다.

타이핑 오류의 횟수는 1

```

GetTickCount() 함수를 사용하여 타이핑 시간을 측정한다. 두 문장을 입력하면서 오타가 발생하면 해당 문자를 \*로 출력하고 문장을 모두 타이핑한 경우, 걸린 시간과 분당 타수, 오타 횟수를 결과로 출력한다.

## 5. 프로세스와 관련된 함수들을 사용하여 프로그램을 작성하여 실행하여 보고, 익숙해지도록 사용해 본다.

### 1) 프로세스 생성 예제 프로그램

#### forkprocess.c

```

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main()
{
    pid_t pid; // 부모에서 프로세스 식별번호 저장
    printf("Calling fork \n");
    pid = fork(); // 새로운 프로세스 생성

    if (pid == 0)
        printf("I'm the child process\n");
    else if (pid > 0)
        printf("I'm the parent process\n");
    else
        printf("fork failed\n");
}

```

### 실행 결과

```
adh@adh: ~/test/SystemProgramming/lab3/no5
adh@adh:~/test/SystemProgramming/lab3/no5$ gedit forkprocess.c
adh@adh:~/test/SystemProgramming/lab3/no5$ gcc -o forkprocess forkprocess.c
adh@adh:~/test/SystemProgramming/lab3/no5$ ./forkprocess
Calling fork
I'm the parent process
I'm the child process
```

fork() 함수를 사용하여 프로세스를 생성한다. 성공적으로 수행되면 호출 프로세스와 똑같은 자식 프로세스가 생성된다.

반환 값으로 부모 프로세스는 자식 프로세스의 ID를 반환하고, 자식 프로세스는 0을 반환한다. forkprocess 파일을 실행하면 부모 프로세스가 자식 프로세스의 ID를 반환하여 "I'm the parent process" 문장이 출력되고, 자식 프로세스가 0을 반환하면서 "I'm the child process" 문장이 출력된다.

## 2) 프로세스 종료 예제 프로그램

### exitprocess.c

```
#include <stdlib.h>
#include <stdio.h>

int main()
{
    int exit_status;
    printf("enter exit status: ");
    scanf("%d", &exit_status);
    exit(exit_status);
}
```

### 실행 결과

```
adh@adh: ~/test/SystemProgramming/lab3/no5
adh@adh:~/test/SystemProgramming/lab3/no5$ gedit exitprocess.c
adh@adh:~/test/SystemProgramming/lab3/no5$ gcc -o exitprocess exitprocess.c
adh@adh:~/test/SystemProgramming/lab3/no5$ ./exitprocess
enter exit status: 0
adh@adh:~/test/SystemProgramming/lab3/no5$ echo $?
0
adh@adh:~/test/SystemProgramming/lab3/no5$ ./exitprocess
enter exit status: 2
adh@adh:~/test/SystemProgramming/lab3/no5$ echo $?
2
```

프로세스 종료는 exit() 함수를 사용한다. status는 종료 상태로, 정상적으로 종료되는 경우는 0, 그렇지 않은 경우는 0이 아닌 값이 반환된다.

## 3) 프로세스 동기화 예제 프로그램

### waitprocess.c

```
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>

int main()
{
    pid_t pid;
```

```

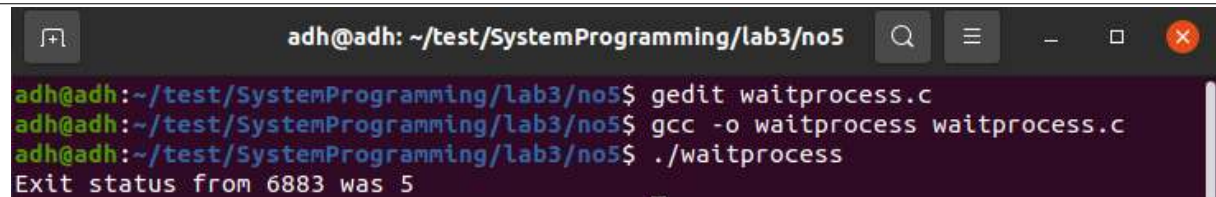
int status, exit_status;
if ((pid = fork()) < 0)
    perror ("fork failed");

if (pid == 0) {
    sleep(4); // 수행을 4초 동안 중단
    exit(5);
}

/* 부모 코드: 자식이 종료될 때까지 대기 */
if ((pid = wait(&status)) == -1) {
    perror("wait failed");
    exit(2);
}
if (WIFEXITED(status)) {
    exit_status = WEXITSTATUS(status);
    printf("Exit status from %d was %d\n", pid, exit_status);
}
exit(0);
}

```

#### 실행 결과



```

adh@adh: ~/test/SystemProgramming/lab3/no5
adh@adh:~/test/SystemProgramming/lab3/no5$ gedit waitprocess.c
adh@adh:~/test/SystemProgramming/lab3/no5$ gcc -o waitprocess waitprocess.c
adh@adh:~/test/SystemProgramming/lab3/no5$ ./waitprocess
Exit status from 6883 was 5

```

wait()는 자식 프로세스들 중 하나가 수행을 마치기를 기다리는 호출로, status는 종료 상태를 의미한다. 자식 프로세스의 종료 상태를 반환하는 WEXITSTATUS 매크로를 사용하여 exit\_status를 구한다.

#### 4) 좀비 프로세스 예제 프로그램

##### zombie.c

```

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

int main(void)
{
    pid_t pid;
    if ((pid = fork()) < 0) {
        perror("fork");
        exit(1);
    }
    else if (pid == 0) // child
        exit(0);
    /* parent */
    sleep(4);
    system("ps -o pid,ppid,state,tty,command");
    exit(0);
}

```



### 실행 결과

```

adh@adh: ~/test/SystemProgramming/lab3/no5
adh@adh:~/test/SystemProgramming/lab3/no5$ gedit zombie.c
adh@adh:~/test/SystemProgramming/lab3/no5$ gcc -o zombie zombie.c
adh@adh:~/test/SystemProgramming/lab3/no5$ ./zombie

```

PID	PPID	S	TT	COMMAND
2086	2076	S	pts/0	bash
6977	2086	S	pts/0	./zombie
6978	6977	Z	pts/0	[zombie] <defunct>
6980	6977	S	pts/0	sh -c ps -o pid,ppid,state,TTY,command
6981	6980	R	pts/0	ps -o pid,ppid,state,TTY,command

부모 프로세스가 wait를 수행하지 않고 있는 상태에서 자식이 퇴장할 때, 퇴장 프로세스는 좀비 프로세스가 된다. 또, 하나 이상의 자식 프로세스가 수행 중인 상태에서 부모가 퇴장할 때, 자식 프로세스는 init 프로세스에게 맡겨지게 된다.

6. system 함수는 쉘 명령이 실행되도록 하는데, 예를 들면, system("ls -al")을 호출 하면, 현재 디렉토리의 파일들을 나열해 준다. 이와 같은 기능을 수행하는 함수를 직접 구현하여 보자. 또, 이 함수를 이용하는 예제 프로그램을 통해서 "a.out ls -al"와 같이 명령이 잘 동작하도록 해 보자.

### systemls.c

```

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

int main(int argc, char* argv[]){
    execl("/bin/ls", argv[1], argv[2], (char *)0);
}

```

### 실행 결과

```

root@9eb578df892c: ~/SystemProgramming/lab3/no6
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no6# ls -al
total 32
drwxr-xr-x 2 root root 4096 Dec 12 13:07 .
drwxr-xr-x 3 root root 4096 Dec 12 13:04 ..
-rwxr-xr-x 1 root root 16912 Dec 12 13:07 systemls
-rw-r--r-- 1 root root 412 Dec 12 13:06 systemls.c
root@9eb578df892c:~/SystemProgramming/lab3/no6# ./systemls ls -la
total 32
drwxr-xr-x 2 root root 4096 Dec 12 13:07 .
drwxr-xr-x 3 root root 4096 Dec 12 13:04 ..
-rwxr-xr-x 1 root root 16912 Dec 12 13:07 systemls
-rw-r--r-- 1 root root 412 Dec 12 13:06 systemls.c

```

execl 함수를 사용하여 argv인자를 char \*로 하나씩 넘겨주도록 한다. 파일을 실행할 때, 명령행 인자로 ls -al을 입력하면 argv[1]에 ls가 들어가고, argv[2]에 -al이 들어간다. 결과로 ls -al 명령의 결과가 출력되는 것을 확인할 수 있다.

## 7. 시그널과 관련된 함수들을 사용하여 프로그램을 작성하고 실행하여 보고, 익숙해 지도록 사용해 본다.

### 1) 시그널 핸들러가 인터럽트 시그널을 받는 프로그램

```
handlesignal.c

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

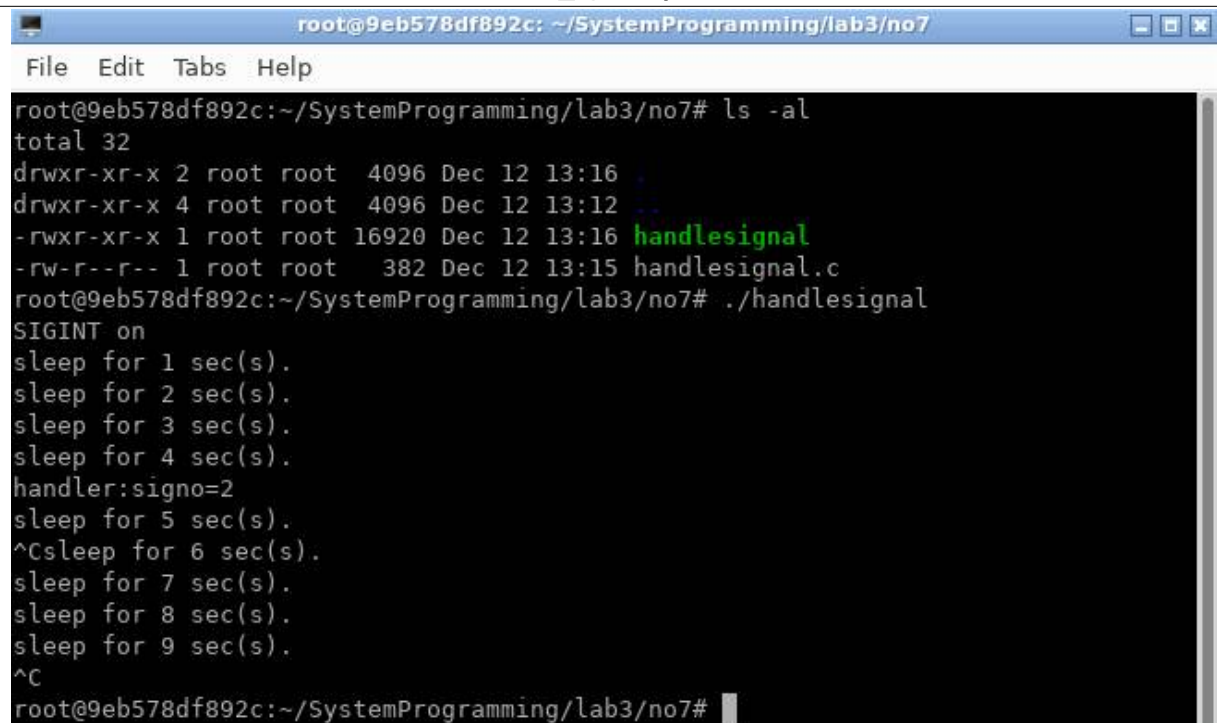
void handler(int signo);

int main(){
    struct sigaction act;
    int i =0;
    act.sa_handler = handler;
    sigfillset(&(act.sa_mask));
    sigaction(SIGINT, &act, NULL);
    printf("SIGINT on\n");

    while(1) {
        sleep(1);
        printf("sleep for %d sec(s).\n", ++i);
    }
}

void handler(int signo){
    printf("handler:signo=%d\n", signo);
}
```

### 실행 결과



```
root@9eb578df892c: ~/SystemProgramming/lab3/no7
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no7# ls -al
total 32
drwxr-xr-x 2 root root 4096 Dec 12 13:16 .
drwxr-xr-x 4 root root 4096 Dec 12 13:12 ..
-rwxr-xr-x 1 root root 16920 Dec 12 13:16 handlesignal
-rw-r--r-- 1 root root 382 Dec 12 13:15 handlesignal.c
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./handlesignal
SIGINT on
sleep for 1 sec(s).
sleep for 2 sec(s).
sleep for 3 sec(s).
sleep for 4 sec(s).
handler:signo=2
sleep for 5 sec(s).
^Csleep for 6 sec(s).
sleep for 7 sec(s).
sleep for 8 sec(s).
sleep for 9 sec(s).
^C
root@9eb578df892c:~/SystemProgramming/lab3/no7#
```

sigaction() 함수를 사용하여 시그널 처리를 결정한다. sigaction 구조체 값을 사용해서 다양한 지정이 가능하다. handlesignal 파일을 실행하면 중간에 Ctrl-C를 입력하였을 때, 인터럽트 시그널이 전달되어 handler에서 "handler:signo=2"를 출력한다.

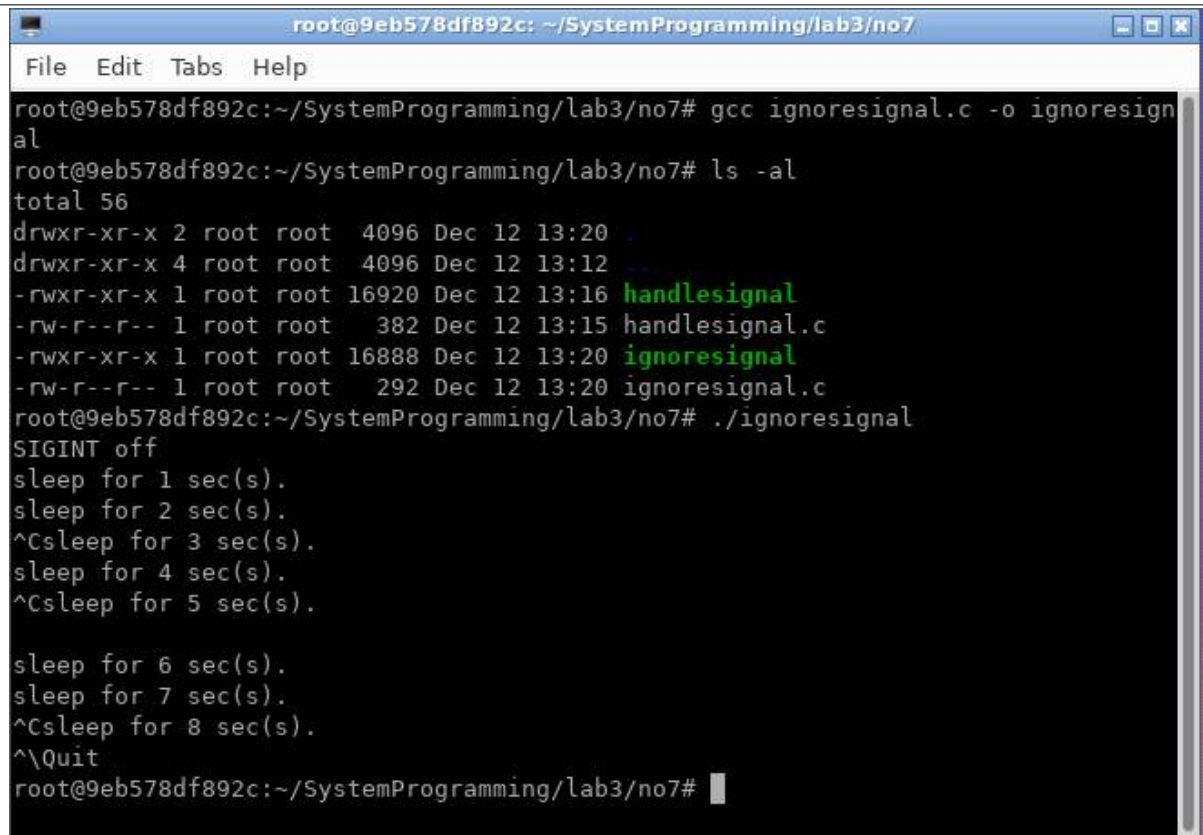
## 2) 시그널을 무시하는 프로그램

### ignoresignal.c

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

int main(){
    struct sigaction act;
    int i =0;
    act.sa_handler = SIG_IGN;
    sigfillset(&(act.sa_mask));
    sigaction(SIGINT, &act, NULL);
    printf("SIGINT off\n");
    while(1) {
        sleep(1);
        printf("sleep for %d sec(s).\n", ++i);
    }
}
```

### 실행 결과



```
root@9eb578df892c: ~/SystemProgramming/lab3/no7
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no7# gcc ignoresignal.c -o ignoresignal
root@9eb578df892c:~/SystemProgramming/lab3/no7# ls -al
total 56
drwxr-xr-x 2 root root 4096 Dec 12 13:20 .
drwxr-xr-x 4 root root 4096 Dec 12 13:12 ..
-rwxr-xr-x 1 root root 16920 Dec 12 13:16 handlesignal
-rw-r--r-- 1 root root 382 Dec 12 13:15 handlesignal.c
-rwxr-xr-x 1 root root 16888 Dec 12 13:20 ignoresignal
-rw-r--r-- 1 root root 292 Dec 12 13:20 ignoresignal.c
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./ignoresignal
SIGINT off
sleep for 1 sec(s).
sleep for 2 sec(s).
^Csleep for 3 sec(s).
sleep for 4 sec(s).
^Csleep for 5 sec(s).

sleep for 6 sec(s).
sleep for 7 sec(s).
^Csleep for 8 sec(s).
^\\Quit
root@9eb578df892c:~/SystemProgramming/lab3/no7#
```

시그널 처리 방법으로 SIG\_IGN을 사용하여 신호를 무시하도록 한다.  
ignoresignal 파일을 실행하면 중간에 Ctrl-C를 입력하여도 인터럽트 신호가 무시되는 것을 확인할 수 있다.

## 3) 시그널을 전송하고 대기하는 프로그램

### sendsignal.c

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

```

int i =0;
void p_handler (int), c_handler (int);

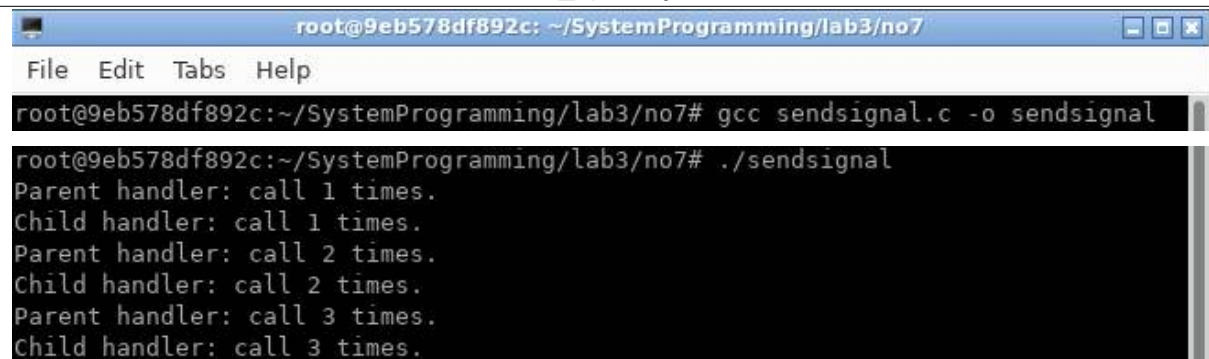
int main(){
    pid_t pid, ppid;
    struct sigaction act;
    pid = fork();
    if (pid == 0) {
        act.sa_handler = c_handler;
        sigaction (SIGUSR1, &act, NULL);
        ppid = getppid(); /* get parent's process id. */
        while (1) {
            sleep (1);
            kill (ppid, SIGUSR1);
            pause();
        }
    } else if (pid >0) {
        act.sa_handler = p_handler;
        sigaction(SIGUSR1, &act, NULL);
        while (1) {
            pause();
            sleep (1);
            kill (pid, SIGUSR1);
        }
    } else
        perror ("Error");
}

void p_handler(int signo) {
    printf("Parent handler: call %d times.\n", ++i);
}

void c_handler(int signo){
    printf("Child handler: call %d times.\n", ++i);
}

```

#### 실행 결과



```

root@9eb578df892c: ~/SystemProgramming/lab3/no7
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no7# gcc sendsignal.c -o sendsignal
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./sendsignal
Parent handler: call 1 times.
Child handler: call 1 times.
Parent handler: call 2 times.
Child handler: call 2 times.
Parent handler: call 3 times.
Child handler: call 3 times.

```

fork() 함수로 프로세스를 생성하여 자식 프로세스의 ID가 반환되면 p\_handler를 통해 "Parent Handler: call 1 times."가 출력되고, 0이 반환되면 c\_handler를 통해 "Child handler: call 1 times."가 출력된다.

#### 4) 자신에게 시그널을 전송하는 프로그램

##### raisesignal.c

```

#include <signal.h>
#include <stdio.h>
#include <stdlib.h>

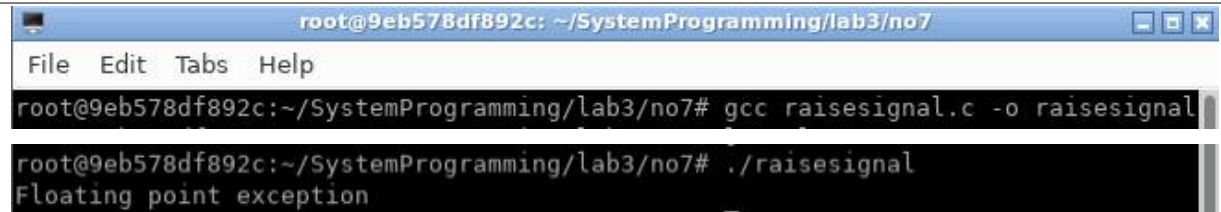
```

```
int main(){
    int a, b;

    a =10;
    b =0;
    if (b==0) /*preempt divide by zero error */
        raise(SIGFPE);
    a = a / b;

    return 0;
}
```

#### 실행 결과



```
root@9eb578df892c: ~/SystemProgramming/lab3/no7
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no7# gcc raisesignal.c -o raisesignal
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./raisesignal
Floating point exception
```

raise() 함수를 사용하여 현재 프로세스에게 시그널을 전송하도록 한다. 파일을 실행하면 a / b에서 b가 0이기 때문에 0으로 나눌 수 없어 오류가 발생한다. 이때, 부동 소수점 예외 로그로 정의된 SIGFPE를 통해 "Floating point exception"이 출력된다.

#### 5) alarm 함수 사용 예제 프로그램

##### alarmsignal.c

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void alarm_handler(int);
int alarm_flag =0;

int main() {
    struct sigaction act;

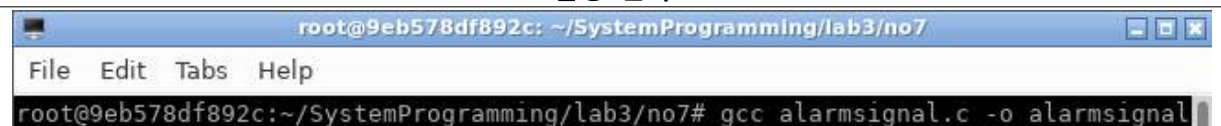
    act.sa_handler = alarm_handler;
    sigaction(SIGALRM, &act, NULL);

    alarm(5);
    pause();
    if (alarm_flag)
        printf("Passed a 5 secs.\n");

    return 0;
}

void alarm_handler(int sig){
    printf("Received a alarm signal. \n");
    alarm_flag =1;
}
```

#### 실행 결과



```
root@9eb578df892c: ~/SystemProgramming/lab3/no7
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no7# gcc alarmsignal.c -o alarmsignal
```

```
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./alarmsignal
Received a alarm signal.
Passed a 5 secs.
```

alarm() 함수를 사용하여 지정된 5초 후에 현재 프로세스에게 시그널을 전달하도록 한다.

## 6) 시그널을 차단하고 해제하는 프로그램

### blocksignal.c

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

int main() {
    sigset_t set1, set2;
    sigfillset(&set1);
    sigemptyset(&set2);
    sigaddset(&set2, SIGINT);

    printf("Critical region start.\n");
    sigprocmask(SIG_BLOCK, &set1, NULL);
    sleep(5);

    printf("Less critical region start.\n");
    sigprocmask(SIG_UNBLOCK, &set2, NULL);
    sleep(5);

    printf("Non critical region start.\n");
    sigprocmask(SIG_UNBLOCK, &set1, NULL);
    sleep(5);

    return 0;
}
```

### 실행 결과

```
root@9eb578df892c: ~/SystemProgramming/lab3/no7
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./blocksignal
Critical region start.
^C
Less critical region start.

root@9eb578df892c:~/SystemProgramming/lab3/no7#
root@9eb578df892c:~/SystemProgramming/lab3/no7# ./blocksignal
Critical region start.
^\\
Less critical region start.
Non critical region start.
Quit
```

sigprocmask() 함수를 사용하여 특정 시그널을 차단하도록 허용한다. SIG\_BLOCK은 차단하는 시그널은 현재 설정된 것들과 set에 포함된 것을 의미하고, SIG\_UNBLOCK은 set에 포함된 시그널들이 차단되지 않도록 설정하는 것이다.



## 8. 프로세스 간 통신 함수들을 사용하여 프로그램을 작성하고 실행하여 보고, 익숙해 지도록 사용해 본다.

### 1) 레코드 잠금 예제 프로그램

```
filelock.c

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

int main()
{
    int fd;
    struct flock testlock;

    int pid;

    testlock.l_type = F_WRLCK;
    testlock.l_whence = SEEK_SET;
    testlock.l_start = 0;
    testlock.l_len = 10;

    /* open file */
    fd = open ("testlock", O_RDWR | O_CREAT, 0666);

    if (fcntl (fd, F_SETLKW, &testlock) == -1) {
        perror ("parent: locking");
        exit (1);
    }

    printf ("parent: locked record\n");

    pid = fork();
    if (pid == 0) { /* child process */
        testlock.l_len = 5;
        if (fcntl (fd, F_SETLKW, &testlock) == -1) {
            perror ("child: locking");
            exit (1);
        }
        printf ("child: locked\n");
        sleep(5);
        printf ("child: exiting\n");
    }
    else if (pid > 0) {
        sleep(5);
        printf ("parent: exiting\n");
    }
    else
        perror ("fork failed");
}
```

### 실행 결과

```
root@9eb578df892c:~/SystemProgramming/lab3/no8# ./filelock
parent: locked record
parent: exiting
child: locked
root@9eb578df892c:~/SystemProgramming/lab3/no8# child: exiting
```

struct flock 구조체를 사용하여 레코드 잠금 기능을 사용할 수 있다. l\_type을 사용하여 잠금 유형을 F\_WRLCK(쓰기 잠금 지정)으로 설정한다. 자식 프로세스가 생성되면 fcntl() 함수를 사용하여 잠금을 구현한다.

## 2) 파이프 예제 프로그램 1

### selfpipe.c

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

#define MSGSIZE 16
char* msg[2] = {"Hello", "World"};

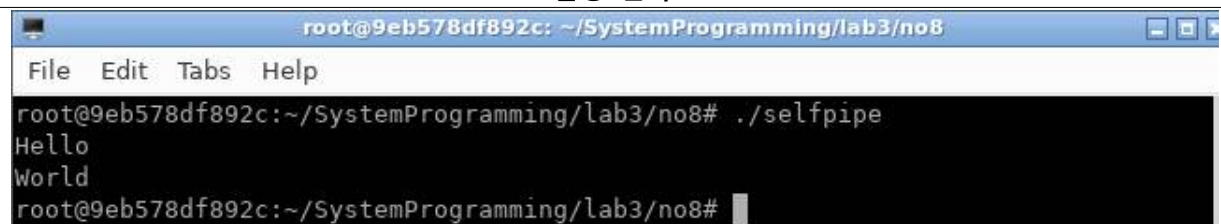
int main() {
    char buf[MSGSIZE];
    int p[2], i;

    /* open pipe */
    if(pipe(p) == -1) {
        perror("pipe call failed");
        exit(1);
    }

    /* write to pipe */
    for(i=0; i<2; i++)
        write(p[1], msg[i], MSGSIZE);

    /* read from pipe */
    for(i=0; i<2; i++) {
        read(p[0], buf, MSGSIZE);
        printf("%s\n", buf);
    }
}
```

### 실행 결과



```
root@9eb578df892c: ~/SystemProgramming/lab3/no8
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no8# ./selfpipe
Hello
World
root@9eb578df892c:~/SystemProgramming/lab3/no8#
```

## 3) 파이프 예제 프로그램 2

### pipetest.c

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

#define MSGSIZE 16

int main()
{
    char buf[MSGSIZE];
    int p[2], i;
    int pid;
```

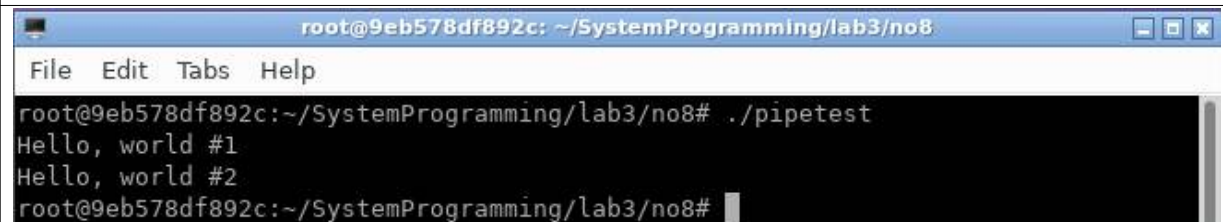
```

/* open pipe */
if (pipe(p) == -1)
{
    perror ("pipe call failed");
    exit(1);
}
pid = fork();
if (pid == 0)    /* child process */
{
    close(p[0]);
    /* write to pipe */
    for (i = 0; i < 2; i++)
    {
        sprintf(buf, "Hello, world #%d", i+1);
        write(p[1], buf, MSGSIZE);
    }
}
else if (pid > 0)
{
    close(p[1]);

    /* read from pipe */
    for (i = 0; i < 2; i++)
    {
        read (p[0], buf, MSGSIZE);
        printf ("%s\n", buf);
    }
}
else    perror ("fork failed");
}

```

#### 실행 결과



```

root@9eb578df892c: ~/SystemProgramming/lab3/no8
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no8# ./pipetest
Hello, world #1
Hello, world #2
root@9eb578df892c:~/SystemProgramming/lab3/no8#

```

#### 4) 비봉쇄 읽기/쓰기 예제 프로그램

##### nonblockpipe.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>

#define MSGSIZE 16

char *parent_name = "parent";
char *child_name = "child";
char *parent_msg = "Hello, child!";
char *child_msg = "Hello, parent!";
void nonblock_rw (char *, int, int, char *);

int main()
{

```

```

int pp[2][2], i;
int pid;

/* open pipe */
for (i = 0; i < 2; i++)
{
    if (pipe(pp[i]) == -1)
    {
        perror ("pipe call failed");
        exit (1);
    }
}

pid = fork();
if (pid == 0) /* child process */
{
    close(pp[0][1]);
    close(pp[1][0]);
    nonblock_rw(child_name, pp[0][0], pp[1][1], child_msg);
}
else if (pid > 0) /* parent process */
{
    close(pp[0][0]);
    close(pp[1][1]);
    nonblock_rw(parent_name, pp[1][0], pp[0][1], parent_msg);
}
else perror ("fork failed");
}

void nonblock_rw (char *name, int read_pipe, int write_pipe, char *message)
{
    char buf[MSGSIZE];
    int nread;
    /* set O_NONBLOCK of fcntl system call */
    if (fcntl (read_pipe, F_SETFL, O_NONBLOCK) == -1)
    {
        perror("read pipe call");
        exit (1);
    }
    if (fcntl (write_pipe, F_SETFL, O_NONBLOCK) == -1)
    {
        perror("write pipe call");
        exit (1);
    }

    while(1)
    {
        switch (nread = read(read_pipe, buf, MSGSIZE))
        {
            case -1:
                if (errno == EAGAIN)
                {
                    printf("%s: pipe empty!\n", name);
                    sleep (1);
                    break;
                }
                else
                {
                    perror("read call");
                    exit (1);
                }
            }
        }
    }
}

```

```

        }
        case 0:
            printf ("%s: read pipe closed\n", name);
            exit (1);
        default:
            printf ("%s: MSG=%s\n", name, buf);
    }
    write(write_pipe, message, MSGSIZE);
    sleep (1);
}
}

```

#### 실행 결과

```

root@9eb578df892c:~/SystemProgramming/lab3/no8# ./nonblockpipe
parent: pipe empty!
child: pipe empty!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
parent: MSG=Hello, parent!
child: MSG=Hello, child!
^C
root@9eb578df892c:~/SystemProgramming/lab3/no8#

```

비봉쇄 읽기/쓰기는 파이프로부터 자료를 읽거나 쓸 때 프로세스가 대기하지 않게 된다.

#### 5) select를 이용한 다중 파이프 예제 프로그램

##### selectpipe.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/wait.h>

#define MSGSIZE 16

char *hello_msg = "Hello, parent!";
char *bye_msg = "Bye, parent!";
void parent(int [3][2]);
void child(int [2]);

int main()
{
    int pp[3][2], i;
    int pid;

```

```

/* open pipe */
for (i =0; i <3; i++)
{
    if (pipe(pp[i]) ==-1)
    {
        perror ("pipe call failed");
        exit (1);
    }
    pid = fork();

    if (pid ==0) child (pp[i]);
    else if (pid ==-1)
    {
        perror ("fork failed");
        exit (1);
    }
}
/* parent process */
parent (pp);
}

void parent(int pp[3][2])
{
    char buf[MSGSIZE], ch;
    fd_set set, master;
    int i;
    for (i =0; i <3; i++)      close(pp[i][1]);

    /* set bit mask of select system call */
    FD_ZERO (&master);
    FD_SET (0, &master);
    for (i =0; i <3; i++)      FD_SET (pp[i][0], &master);
    while (set=master, select (pp[2][0]+1, &set, NULL, NULL, NULL) >0)
    {
        if (FD_ISSET(0, &set))
        {
            printf("From standard input: \n");
            read (0, &ch, 1);
            printf("%c\n", ch);
        }
        for (i =0; i <3; i++)
        {
            if (FD_ISSET(pp[i][0], &set) && read(pp[i][0], buf, MSGSIZE) >0)
                printf("message is %s from child %d\n", buf, i);
        }
        if (waitpid (-1, NULL, WNOHANG) ==-1)
            return;
    }
}

void child(int p[2])
{
    int j;

    /* write to pipe */
    close (p[0]);
    for (j =0; j <3; j++)
    {
        write(p[1], hello_msg, MSGSIZE);
    }
}

```



```

        sleep (getpid() % 4);
    }

    write(p[1], bye_msg, MSGSIZE);
    exit (0);
}

```

#### 실행 결과

```

root@9eb578df892c: ~/SystemProgramming/lab3/no8# ./selectpipe
message is Hello, parent! from child 0
message is Hello, parent! from child 1
message is Hello, parent! from child 2
message is Hello, parent! from child 0
message is Hello, parent! from child 0
message is Hello, parent! from child 1
a
From standard input:
a
From standard input:
message is Hello, parent! from child 2
message is Bye, parent! from child 0
message is Hello, parent! from child 1
message is Hello, parent! from child 2
message is Bye, parent! from child 1
message is Bye, parent! from child 2

```

select() 함수는 다중 입출력 관리 기능으로, 지정된 파일 기술자 집합 중 읽기와 쓰기가 준비된 것이 있는지 또는 오류가 있는지 등을 검사할 수 있다.  
select() 함수의 첫 번째 인자로 검사할 파일 기술자 번호 최대값+1인 pp[2][0]+1을 입력한다. 두 번째 인자로 파일 기술자를 set 집합에 추가하도록 지정한다.

## 9. 메시지 큐를 사용하여 텍스트 기반의 간단한 채팅 프로그램을 구현하시오.

#### sendmsg.c

```

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

#define BUFSIZE 16
#define QKEY (key_t)0111
struct msgq_data {
    long type;
    char text[BUFSIZE];
};

struct msgq_data send_data = {1, "Hi gitae~!!"};

int main()
{
    int qid, len;
    char buf[BUFSIZE];
    if ((qid = msgget(QKEY, IPC_CREAT | 0666)) == -1) {

```

```

    perror ("msgget failed");
    exit (1);
}
if(msgsnd(qid,&send_data,strlen(send_data.text),0)==-1) {
    perror ("msgsnd failed");
    exit (1);
}
}

```

#### receivemsg.c

```

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

#define BUFSIZE 16
#define QKEY (key_t)0111
struct msgq_data {
    long type;
    char text[BUFSIZE];
};

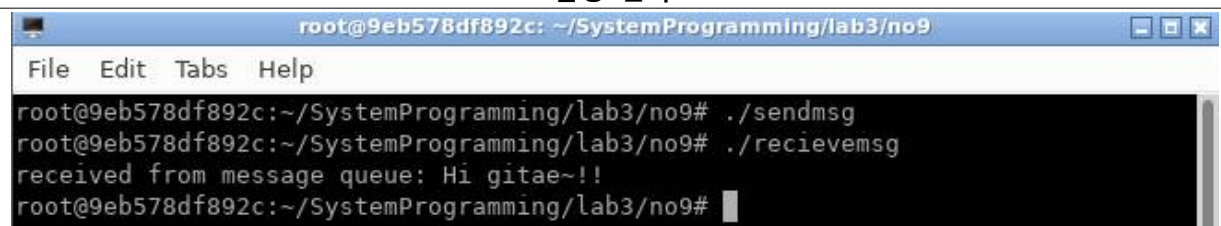
struct msgq_data rcv_data;
int main()
{
    int qid, len;
    if ((qid = msgget(QKEY, IPC_CREAT | 0666)) == -1) {
        perror ("msgget failed");
        exit (1);
    }

    if ((len=msgrcv(qid, &rcv_data, BUFSIZE, 0, 0)) == -1) {
        perror ("msgrcv failed");
        exit (1);
    }

    printf("received from message queue: %s\n",
        rcv_data.text);
    if (msgctl(qid, IPC_RMID, 0) == -1) {
        perror ("msgctl failed");
        exit (1);
    }
}

```

#### 실행 결과



```

root@9eb578df892c: ~/SystemProgramming/lab3/no9
File Edit Tabs Help
root@9eb578df892c:~/SystemProgramming/lab3/no9# ./sendmsg
root@9eb578df892c:~/SystemProgramming/lab3/no9# ./recievemsg
received from message queue: Hi gitae~!!
root@9eb578df892c:~/SystemProgramming/lab3/no9#

```

msgget() 함수를 사용하여 System V의 메시지 큐 id를 얻어온다. 보낼 메시지로 "Hi gitae~!!" 문자를 담아서 msgsnd() 함수를 통해 메시지를 보낸다. 메시지를 받을 때는 msgrcv() 함수를 사용한다. 메시지 큐를 제어하기 위한 시스템 콜로는 msgctl() 함수를 사용한다. IPC\_RMID는 메시지 큐를 지우는 것을 의미한다.

10. 공유 메모리를 사용하여 한 파일을 다른 파일로 복사하는 프로그램을 작성하시오. 단, 부모(읽는 프로세스)와 자식(쓰는 프로세스) 프로세스가 공유 메모리 영역을 동시에 접근하는 일이 없도록 세마포어 같은 동기화 기법을 활용하시오.

copy\_file.c

```
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/mman.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <semaphore.h>
#include <errno.h>
#include <unistd.h>

#define SHMSIZE 4096
#define SHMNAME "/my_shm"
#define SEMNAME "/my_sem"

void p(sem_t *semd);
void v(sem_t *semd);

void main(int argc, char* argv[])
{
    FILE *fp, *copy_fp; // file ptr
    char *path = argv[1];
    char buf[SHMSIZE];
    int status, shmd, count;
    void *shmaddr;
    pid_t pid;
    sem_t *semd;
    int i, val;
    if ((semd = sem_open(SEMNAME, O_CREAT, 0600, 1)) == SEM_FAILED) {
        perror ("sem_open failed");
        exit (1);
    } // semaphore Open
    sem_getvalue(semd, &val);
    memset(buf, 0, SHMSIZE); // memory init
    if((pid = fork()) == 0){
        /* child shm write*/
        if ((shmd = shm_open(SHMNAME, O_CREAT | O_RDWR, 0666)) == -1) {
            perror ("shm_open failed");
            exit (1);
        }
        if (ftruncate(shmd, SHMSIZE) != 0) {
            perror ("ftruncate failed");
            exit (1);
        }
        if ((shmaddr = mmap(0, SHMSIZE, PROT_WRITE, MAP_SHARED, shmd, 0)) ==
MAP_FAILED) {
            perror ("mmap failed");
            exit (1);
        }
        if((fp = fopen(path, "r")) == NULL){ // file exception handle
            perror("fopen");
            exit(1);
        }
    }
```

```

}
if (mlock(shmaddr, SHMSIZE) !=0) {
    perror ("mlock failed");
    exit (1);
}

p(sem);
printf("Child Process %d : semaphore in use\n", getpid());
while(feof(fp) !=0){
    count = fread(buf, sizeof(char), SHMSIZE, fp);
    strcpy((char *)shmaddr, buf);
}
v(sem);
printf("Child Process %d : putting semaphore\n", getpid());

if (munmap(shmaddr, SHMSIZE) ==-1) {
    perror ("munmap failed");
    exit (1);
}
fclose(fp);
close(shmd);
exit(0);
} else if (pid >0){
    /* parents read */
    pid = wait(&status);

    if ((shmd = shm_open(SHMNAME, O_RDWR, 0666)) ==-1) {
        perror ("shm_open failed");
        exit (1);
    }

    if ((shmaddr = mmap(0, SHMSIZE, PROT_READ, MAP_SHARED, shmd, 0)) == MAP_FAILED)
    {
        perror ("mmap failed");
        exit (1);
    }

    if (mlock(shmaddr, SHMSIZE) !=0) {
        perror ("mlock failed");
        exit (1);
    }
    strcat(path, "_copy"); // append path + "_copy"
    if((copy_fp = fopen(path, "w")) ==NULL){ // file exception handle
        perror("fopen");
        exit(1);
    }
    p(sem);
    printf("Parent Process %d : semaphore in use\n", getpid());
    // shm data copy to copy_fp
    fwrite(shmaddr, sizeof(char), SHMSIZE, copy_fp);
    fclose(copy_fp);

    if (munmap(shmaddr, SHMSIZE) ==-1) {
        perror ("munmap failed");
        exit (1);
    }
    printf("Parents Process %d : putting semaphore\n", getpid());
    v(sem);
    close(shmd);
}

```

```

    if (shm_unlink(SHMNAME) == -1) {
        perror ("shm_unlink failed");
        exit (1);
    }
    if (sem_close(semid) == -1) {
        perror ("sem_close failed");
        exit (1);
    }
} else {
    perror("fork Error");
    exit(1);
}
}
void p(sem_t *semid) {
    int ret;
    if ((ret = sem_trywait(semid)) != 0 && errno == EAGAIN)
        sem_wait(semid);
    else if (ret != 0) {
        perror ("sem_trywait failed");
        exit (1);
    }
}
void v(sem_t *semid) {
    if (sem_post(semid) != 0) {
        perror ("sem_post failed");
        exit (1);
    }
}
}

```

#### 실행 결과

```

adh@adh:~/test/SystemProgramming/lab3/no10$ gcc copy_file.c -o copy_file -lrt -lpthread
adh@adh:~/test/SystemProgramming/lab3/no10$ ./copy_file copy_file.c
Child Process 8094 : semaphore in use
Child Process 8094 : putting semaphore
Parent Process 8093 : semaphore in use
Parents Process 8093 : putting semaphore

```

공유 메모리는 프로세스와 프로세스 사이의 공유할 수 있는 메모리 공간을 지정하는 방법이다. 공유 메모리를 사용하기 위해서는 공유 메모리 객체 생성 -> 공유 메모리 객체의 크기 설정 -> 공유 메모리 객체를 프로세스 메모리 영역과 매핑하는 과정이 필요하다.

shm\_open() 함수는 특정 이름을 갖는 공유 메모리 객체를 생성한 후에 그 객체에 접근할 수 있는 파일 디스크립터를 반환한다. 두 번째 매개변수로, O\_CREATE 매크로를 입력하여 해당 이름의 객체가 존재하지 않을 경우 새로운 객체를 생성하도록 한다. (O\_RDWR은 객체를 읽기, 쓰기가 가능한 상태로 설정)

생성한 공유 메모리의 크기를 지정하기 위해서는 ftruncate() 함수를 사용한다.

매핑하는 과정에는 mmap() 함수를 사용한다.

세 번째 매개변수로 매핑 영역의 보호 수준을 지정한다. PROT\_READ는 데이터의 읽기 작업이 가능하도록 하는 것이고, PROT\_WRITE는 데이터의 쓰기 작업이 가능하도록 하는 매크로이다.

네 번째 매개변수는 매핑된 데이터의 처리 방식에 대한 정보를 지정하며 MAP\_SHARED는 다른 프로세스와 매핑된 영역을 공유하는 것을 의미한다.

공유 메모리 영역의 매핑을 제거할 때는 munmap() 함수를 사용한다.

공유 메모리는 정보의 갱신 여부를 다른 프로세스에게 알리는 작업과 공유 메모리 영역에 대한 프로세스 간의 동기화 작업이 필요하다. 이를 위해 동기화 방법 중 하나인 세마포어를 사용한다. 세마포어를 사용하기 위해 sem\_t \*sem\_open(const char \*name, int oflag, mode\_t mode, unsigned int value)의 형식으로 객체를 생성한다.

`sem_getvalue()`는 `sval`이 가리키는 위치에 `sem` 세마포어의 현재 값을 저장한다.

`sem_wait()` 함수를 사용하여 세마포어의 값을 확인하고 세마포어의 값이 0인 경우 그 값이 양수가 될 때까지 프로세스를 대기 상태로 전환시킨다.

`sem_trywait()` 함수는 `sem_wait()`의 비봉쇄 버전으로, `sem`이 가리키는 세마포어의 값이 0이 아니면 카운터 값이 감소되며 `sem_trywait`은 즉시 0을 반환한다.

크리티컬 섹션에 진입한 프로세스가 코드를 실행한 후에는 다른 프로세스가 크리티컬 섹션에 진입할 수 있도록 `sem_post()` 함수를 통해 자원 반납을 진행한다.