Soongsil University
School of Computer Science and Engineering
(The Best CSE in Korea)

학번 (Student ID)	
이름 (Student Name)	

Course Title	**Linux System Programming Design and Lab.**
Instructor	**Jiman Hong (jiman@ssu.ac.kr)**
Date of Exam	**April 28, 2017**
Duration of Exam	**5 hours**
Number of Exam Page (including this cover page)	**37**
Number of Question	**49 + Bonus 1**
Additional Materials Allowed	**None**
Grade	

- Write your answers in exam sheets!
- Write only to the space provide for each question!
- Remember to print your name clearly on the exam book cover!
- Show Your Work and Reasoning Clearly!
- Don't cheat. This is a closed-book exam!
- Good Luck!

※ 다음 물음에 답하시오. 단, 틀린 설명을 찾는 문제에서 틀린 설명은 여러 개일 수 있으며, 틀린 설명이 없을 경우 답없음에 답을 표하시오. [1~10, 각 0.5점]

1. creat() 함수를 open() 함수로 표현하시오.

2. rewind() 함수를 fseek() 함수로 표현하시오.

3. 다음 unlink() 함수와 remove() 함수에 대한 설명이 틀린 것을 고르시오.

(1) unlink는 파일에 대한 디렉토리 항목을 지우고 링크 개수를 증가시킨다.

(2) unlink는 모든 디렉토리에 대해 쓰기 및 실행 권한이 필요하다.

(3) rm 명령어의 경우 remove를 사용한다.

(4) remove는 대상이 파일이면 unlink와 동일하다.

(5) 답 없음

4. 다음 심볼릭 링크와 하드 링크에 대한 설명이 틀린 것을 고르시오.

(1) 심볼릭 링크는 파일에 대한 간접 포인터이다.

(2) 디렉토리에 대한 하드 링크는 슈퍼유저만이 만들 수 있다.

(3) 심볼릭 링크에서는 링크와 해당 파일이 다른 파일시스템에 존재해도 된다.

(4) 심볼릭 링크는 주로 일반 파일을 링크할 때 사용한다.

(5) 심볼릭 링크는 원래 파일이 삭제되면 실제 데이터 블록을 참조하던 inode도 삭제된다.

(6) 답 없음

5. 다음 dup과 dup2 함수에 대한 설명이 틀린 것은?

(1) dup가 돌려주는 새 파일 디스크립터는 사용 가능한 파일 디스크립터들 중 가장 낮은 번호임이 보장된다.

(2) dup2의 경우 filedes2 인수로 지정된 값이 새 파일 디스크립터의 값이 된다.

(3) dup2의 경우 filedes2가 이미 열려 있다면 먼저 그것을 닫은 후에 복제한다.

(4) dup2의 경우 filedes가 filedes2와 같으면 filedes2을 닫고 리턴한다.

(5) 답 없음

6. 다음 umask 함수에 대한 설명이 틀린 것은?

(1) 파일 생성 마스크는 시스템의 보안장치의 하나이다.

(2) mode 인수의 비트들 중 파일 생성 마스크에서 켜 있는 비트들은 파일 생성 시 자동으로 권한이 꺼진다.

(3) umask는 <sys/stat.h>에 정의된 상수들 중 하나 또는 여러 개를 비트 단위로 and 연산으로 결합한다.

(4) umask는 <sys/stat.h>에 정의된 상수들 중 그룹의 실행은 S_IXGRP이다.

(5) 답 없음

7. 다음 read 함수에서 실제로 읽은 바이트 수(반환값)가 호출시 요청한 바이트 수(nbytes 인수)보다 작은 경우의 설명으로 틀린 것은?

(1) 정규 파일을 읽을 때, 요청된 수만큼의 바이트들을 읽기 전에 파일의 끝에 도달

(2) 터미널 파일에서 자료를 읽을 때

(3) 네트워크에서 자료를 읽을 때

(4) 파이프나 FIFO에서 자료를 읽을 때

(5) 레코드 기반 장치에서 자료를 읽을 때

(6) 답 없음

8. 다음 디렉토리의 내용을 보기 위한 함수에 대한 설명이 틀린 것은?

(1) DIR 구조체는 읽어 들이고 싶은 디렉토리의 정보를 처리하는데 필요한 시스템 내부 구조체로서 FILE 구조체와 유사한 방법으로 처리된다.

(2) opendir의 리턴값으로 얻을 수 있는 DIR 구조체에 대한 포인터는 나머지 함수들의 인자로 쓰인다.

(3) opendir은 readdir에 의해 첫 번째 항목이 읽혀지도록 시스템 내부 구조체를 초기화하는 역할을 한다.

(4) 읽혀지는 항목의 순서는 시스템에 따라 다를 수 있으며, 보통 알파벳 순서를 따른다.

(5) 답 없음

9. 다음 setbuf 함수와 setvbuf 함수에 대한 설명이 틀린 것은?

(1) 표준 I/O 라이브러리가 제공하는 버퍼링의 목표는 read 호출과 write 호출의 횟수를 최소화 하는 것이다.

(2) 표준 I/O 라이브러리가 제공하는 전체 버퍼링은 표준 I/O 버퍼가 꽉 찼을 때 실제 I/O가 일어나며 전체 버퍼링에 사용되는 버퍼는 주어진 한 스트림에 대하여 I/O가 처음 수행 될 시 malloc을 호출해서 할당된다.

(3) 표준 I/O 라이브러리가 제공하는 줄 단위버퍼링은 입력이나 출력에서 새줄 문자를 만났을 때 I/O를 수행하며 주로 터미널을 가리키는 스트림에 쓰인다.

(4) 표준 I/O 라이브러리가 제공하는 버퍼링 없음 기능은 문자들을 즉시 버퍼링한다.

(5) setvbuf를 통하여 버퍼를 켜거나 끌 수 있으며 버퍼링을 켤 경우, stdio.h에 정의된 상수인 BUFSIZ 크기를 가리키는 포인터를 buf 인수에 지정해야 한다.

(6) 답 없음

10. 다음 access 함수에 대한 설명이 틀린 것은?

(1) access 함수는 파일에 사용자가 접근할 수 있는지에 대한 판정을 하는 함수이다.

(2) access 함수는 실제 사용자 ID와 실제 그룹 ID에 기초해서 파일에 대한 권한을 판정한다.

(3) access 함수의 mode 인수에 쓰이는 상수 중 R_OK는 읽기 권한을 판정한다.

(4) access 함수의 mode 인수에 쓰이는 상수 중 X_OK는 실행 권한을 판정한다.

(5) 답 없음

※ 다음 물음에 답하시오. [11~45]

11. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 잘못된 부분을 고치시오. [2.5점]

<ssu_test.txt>
Linux System Programming! Unix System Programming! Linux Mania Unix Mania
<ssu_open.c>
#include <stdio.h> #include <stdlib.h>

```
#include <fcntl.h>

int main(void)
{
    char fname = 'ssu_test.txt';
    int fd;

    if ((fd = open(fname, O_RDONLY)) < 0) {
        fprintf(stderr, "open error for %d\n", fname);
        exit(0);
    }
    else
        printf("Success!\nFilename : %s\nDescriptor : %d\n", fname, fd);

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_open
Success!
Filename : ssu_test.txt
Descriptor : 3
```

12. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

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

int main(void)
{
    char *fname = "ssu_test.txt";
    int fd;

    if ((fd = creat(          , 0666)) < 0) {
        fprintf(          , "creat error for %s\n", fname);
        exit(1);
    }
    else {
        printf("Success!\nFilename : %s\nDescriptor : %d\n", fname,          );
                ;
    }

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_creat
Success!
Filename : ssu_test.txt
```

Descriptor : 3

13. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2점]

<ssu_employee.h>

```
#define NAME_SIZE 64

struct ssu_employee {
    char name[NAME_SIZE];
    int pid;
    int salary;
};
```

<ssu_read.c>

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include "ssu_employee.h"

int main(int argc, char *argv[])
{
    struct ssu_employee record;
    int fd;
    int record_num;

    if (argc < [   ] ){
        fprintf(stderr, "Usage : %s file\n", argv[0]);
        exit(1);
    }

    if ((fd = open(argv[1], O_RDONLY)) < 0) {
        fprintf(stderr, "open error for %s\n", argv[1]);
        exit(1);
    }

    while (1) {
        printf("Enter record number : ");
        scanf("%d", [          ]);

        if (record_num < 0)
            break;

        if (lseek([                              ]) < 0) {
            fprintf(stderr, "lseek error\n");
            exit(1);
        }

        if (read([                              ]) > 0)
            printf("Employee : %s Salary : %d\n", record.name, record.salary);
        else
            printf("Record %d not found\n", record_num);
```

```
    }

    close(fd);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_open
Success!
Filename : ssu_test.txt
Descriptor : 3
```

14. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

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

#define BUFFER_SIZE 1024

int main(void)
{
    char buf[BUFFER_SIZE];
    char *fname = "ssu_test.txt";
    int count;
    int fd1, fd2;

    if ((fd1 = open(fname, O_RDONLY, 0644)) < 0) {
        fprintf(stderr, "open error for %s\n", fname);
        exit(1);
    }
    fd2 = dup(          );
    count = read(                  );
    buf[count] = 0;
    printf("fd1's printf : %s\n", buf);
    lseek(                          );
    count = read(          );
    buf[count] =        ;
    printf("fd2's printf : %s\n", buf);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_dup
fd1's printf : Linux System
fd2's printf : Programming!
```

15. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 잘못된 부분을 고치시오. [2.5점]

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

```
#include <type.h>

int main(int argc, char *argv[]) {
    struct stat statbuf;

    if (argc != 2) {
        fprintf (stderr, "usage: %s <file>\n", argv[0]);
        exit(1);
    }

    if ((stat(argv[2], statbuf)) < 0 ) {
        fprintf(stderr, "stat error\n");
        exit(1);
    }

    printf("%s is %ld bytes\n", argv[1], statbuf.stat_size);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_stat ssu_test.txt
ssu_test.txt is 74 bytes
```

16. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [4.5점]

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

int main(int argc, char *argv[])
{
    struct stat file_info;
    char *str;
    int i;

    for (i = 1; i < argc; i++) {
        printf("name = %s, type = ", argv[i]);

        if (lstat(                              ) < 0) {
            fprintf(stderr, "lstat error\n");
            continue;
        }

        if (                                    )
            str = "regular";
        else if (                               )
            str = "directory";
        else if (                               )
            str = "character special";
        else if (                               )
```

```
            str = “block special”;
        else if (                                    )
            str = “FIFO”;
        else if (                                    )
            str = “symbolic link”;
        else if (                                    )
            str = “socket”;
        else
            str = “unknown mode”;

        printf(“%s\n”,                  );
    }
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_file_macro /etc/passwd /home /dev/sda
name = /etc/passwd, type = regular
name = /home, type = directory
name = /dev/sda, type = block special
```

17. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2점]

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

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

    if (argc < 2) {
        fprintf(stderr, "usage: %s <file1> <file2> .. <fileN>\n", argv[0]);
        exit(1);
    }

    for (i = 1; i < argc; i++) {
        if (access(argv[i],       ) < 0) {
            fprintf(stderr, "%s doesn't exist.\n", argv[i]);
            continue;
        }

        if (access(argv[i],       ) == 0)
            printf("User can read %s\n", argv[i]);

        if (access(argv[i],       ) == 0)
            printf("User can write %s\n", argv[i]);

        if (access(argv[i],       ) == 0)
            printf("User can execute %s\n", argv[i]);
    }
```

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

실행결과

```
root@localhost:/home/oslab# ./ssu_access /home/oslab oslab
User can read /home/oslab
User can write /home/oslab
User can execute /home/oslab
oslab doesn't exist.
```

18. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include [          ]

#define RM_MODE (S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP|S_IROTH|S_IWOTH)

int main(void)
{
    char *fname1 = "ssu_file1";
    char *fname2 = "ssu_file2";

    [                    ];

    if (creat([                ]) < 0) {
        fprintf(stderr,"creat error for %s\n",fname1);
        exit(1);
    }
    umask([                    ]);

    if (creat([                ]) < 0) {
        fprintf(stderr,"creat error for %s\n",fname2);
        exit(1);
    }

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# umask 002
root@localhost:/home/oslab# umask
0002
root@localhost:/home/oslab# ./ssu_umask
root@localhost:/home/oslab# ls -l ssu_file1 ssu_file2
-rw-rw-rw- 1 root root 0 Jan  8 22:09 ssu_file1
-rw------- 1 root root 0 Jan  8 22:09 ssu_file2
root@localhost:/home/oslab# umask
0002
```

19. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

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

int main(void)
{
    [          ]statbuf;
    char *fname1 = "ssu_file1";
    char *fname2 = "ssu_file2";

    if (stat([          ]) < 0)
        fprintf(stderr, "stat error %s\n", fname1);

    if (chmod([          ]) < 0)
        fprintf(stderr, "chmod error %s\n", fname1);

    if (chmod([          ]) < 0)
        fprintf(stderr, "chmod error %s\n", fname2);

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ls -l ssu_file1 ssu_file2
-rw-rw-rw- 1 root root 0 Jan  8 22:09 ssu_file1
-rw------- 1 root root 0 Jan  8 22:09 ssu_file2
root@localhost:/home/oslab# ./ssu_chmod
root@localhost:/home/oslab# ls -l ssu_file1 ssu_file2
-rwSrw-rw- 1 root root 0 Jan  8 22:09 ssu_file1
-rw-r--r-x 1 root root 0 Jan  8 22:09 ssu_file2
```

20. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

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

int main(void)
{
    struct [          ] statbuf;
    char *fname = "ssu_myfile";
    int fd;

    if ((fd = open(fname, O_RDWR | O_CREAT, 0600)) < 0) {
        fprintf(stderr, "open error for %s\n", fname);
        exit(1);
    }
    close(fd);
```

```
    stat(                              );
    printf("# 1st stat call # UID:%d     GID:%d\n", statbuf.st_uid, statbuf.st_gid);

    if (chown(                         ) < 0) {
        fprintf(stderr, "chown error for %s\n", fname);
        exit(1);
    }

    stat(                              );
    printf("# 2nd stat call # UID:%d     GID:%d\n", statbuf.st_uid, statbuf.st_gid);

    if (unlink(                ) < 0) {
        fprintf(stderr, "unlink error for %s\n", fname);
        exit(1);
    }
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_chown
# 1st stat call # UID:0     GID:0
# 2nd stat call # UID:501     GID:100
```

21. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

<ssu_oslab.c>

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

int main(void)
{
    printf("This is oslab file\n");
    exit(0);
}
```

<ssu_link.c>

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

int main(int argc, char *argv[])
{
    if (argc < 2) {
        fprintf(stderr, "usage: %s <file1> <file2>\n", argv[0]);          exit(1);
    }
    if (link(                          ) == -1) {
        fprintf(stderr, "link error for %s\n", argv[1]);
        exit(1);
    }
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# gcc -o oslab ssu_oslab.c
```

```
root@localhost:/home/oslab# ./ssu_link oslab eoslab
root@localhost:/home/oslab# ls -l oslab eoslab
-rwxr-xr-x [          ] root root 8656 Jan  8 23:16 eoslab
-rwxr-xr-x [          ] root root 8656 Jan  8 23:16 oslab
root@localhost:/home/oslab# ./oslab
[                                  ]
root@localhost:/home/oslab# ./eoslab
[                                  ]
```

22. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2.5점]

<ssu_dump.txt>

```
Linux System Programming!
```

<ssu_unlink.c>

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

int main(void)
{
    char *fname = "ssu_dump.txt";
    if (open([                    ]) < 0) {
        fprintf(stderr, "open error for %s\n", fname);
        exit(1);
    }
    if (unlink([                  ]) < 0) {
        fprintf(stderr, "unlink error for %s\n", fname);
        exit(1);
    }
    printf("File unlinked\n");
    sleep(20);
    printf("Done\n");
    exit(0);
}
```

실행결과 [Ubuntu, Fedora], [일반 사용자 권한으로 실행]

```
oslab@localhost:~$ ls -l ssu_dump.txt
-rw-rw-r-- 1 oslab oslab 26 Jan  9 23:06 ssu_dump.txt
oslab@localhost:~$ [        ] /home/
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1       16381864 4253648  11273024  28% /
oslab@localhost:~$ ./ssu_unlink &
[1] 3970
oslab@localhost:~$ File unlinked

oslab@localhost:~$ ls -l ssu_dump.txt
ls: cannot access 'ssu_dump.txt': No such file or directory
oslab@localhost:~$ [        ] /home/
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1       16381864 4253648  11273024  28% /
```

```
oslab@localhost:~$ Done

[1]+  Done                    ./ssu_unlink
oslab@localhost:~$ [          ] /home/
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda1       16381864 4253644  11273028  28% /
```

23. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1점]

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

int main(int argc, char *argv[])
{
    if (argc != 3) {
        fprintf(stderr, "usage: %s <oldname> <newname>\n",argv[0]);
        exit(1);
    }

    if (link([                          ]) < 0) {
        fprintf(stderr, "link error\n");
        exit(1);
    }
    else
        printf("step1 passed.\n");

    if (remove(argv[1]) < 0) {
        fprintf(stderr, "remove error\n");
        remove([                          ]);
        exit(1);
    }
    else
        printf("step2 passed\n");

    printf("Success!\n");
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# vi ssu_abcd.txt
root@localhost:/home/oslab# ls -l ssu_abcd.txt
-rw-r--r-- 1 root root 0 Jan  8 23:34 ssu_abcd.txt
root@localhost:/home/oslab# ./ssu_remove ssu_abcd.txt ssu_efgh.txt
step1 passed.
step2 passed.
Success!
root@localhost:/home/oslab# ls ssu_abcd.txt
ls: cannot access 'ssu_abcd.txt': No such file or directory
root@localhost:/home/oslab# ls -l ssu_efgh.txt
-rw-r--r-- 1 root root 0 Jan  8 23:34 ssu_efgh.txt
```

24. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [3점]

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

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

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

    if ((fd = open(argv[1], O_RDONLY)) < 0) {
        fprintf(stderr, "first open error for %s\n", argv[1]);
        exit(1);
    }
    else
        close(fd);

    if (rename([                    ]) < 0) {
        fprintf(stderr, "rename error\n");
        exit(1);
    }
    // 읽기 전용으로 open()
    if ((fd = open([                    ])) < 0)
        printf("second open error for %s\n", [          ]);
    else {
        fprintf(stderr, "it's very strange!\n");
        exit(1);
    }
    // 읽기 전용으로 open()
    if ((fd = open([                    ])) < 0) {
        fprintf(stderr, "third open error for %s\n", [          ]);
        exit(1);
    }
    printf("Everything is good!\n");
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# vi ssu_test1.txt
root@localhost:/home/oslab# ./ssu_rename ssu_test1.txt ssu_test2.txt
second open error for ssu_test1.txt
Everything is good!
root@localhost:/home/oslab# ls ssu_test2.txt
ssu_test2.txt
```

25. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [4점]

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

int main(int argc, char *argv[])
{
    struct [          ] time_buf;
    struct [          ] statbuf;
    int fd;
    int i;

    for (i = 1; i < argc; i++) {
        if (stat([          ]) < 0) {
            fprintf(stderr, "stat error for %s\n", argv[i]);
            continue;
        }
        if ((fd = open([          ])) < 0) {
            fprintf(stderr, "open error for %s\n", argv[i]);
            continue;
        }
        close(fd);
        time_buf.actime  = [          ];
        time_buf.modtime = [          ];
        if (utime([          ]) < 0) {
            fprintf(stderr, "utime error for %s\n", argv[i]);
            continue;
        }
    }
    exit(0);
}
```

실행결과

```
oslab@localhost:~$ vi ssu_test1.txt
oslab@localhost:~$ vi ssu_test2.txt
oslab@localhost:~$ ls -l ssu_test1.txt ssu_test2.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:15 ssu_test1.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:16 ssu_test2.txt
oslab@localhost:~$ ls -lu ssu_test1.txt ssu_test2.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:15 ssu_test1.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:16 ssu_test2.txt
oslab@localhost:~$ date
Mon Jan  9 00:17:43 PST 2017
oslab@localhost:~$ ./ssu_utime ssu_test1.txt ssu_test2.txt
oslab@localhost:~$ ls -l ssu_test1.txt ssu_test2.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:15 ssu_test1.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:16 ssu_test2.txt
oslab@localhost:~$ ls -lu ssu_test1.txt ssu_test2.txt
```

```
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:15 ssu_test1.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:16 ssu_test2.txt
oslab@localhost:~$ ls -lc ssu_test1.txt ssu_test2.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:17 ssu_test1.txt
-rw-rw-r-- 1 oslab oslab 0 Jan  9 00:17 ssu_test2.txt
```

26. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [5점]

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include [          ]
#include <fcntl.h>
#include <string.h>
#include [          ]

#define DIRECTORY_SIZE MAXNAMLEN

int main(int argc, char *argv[]) {
    struct [          ] *dentry;
    struct [          ]  statbuf;
    char filename[DIRECTORY_SIZE + 1];
    DIR  [          ];

    if(argc < 2) {
        fprintf(stderr, "usage: %s <directory>\n", argv[0]);
        exit(1);
    }
    if((dirp = opendir(argv[1])) == NULL || chdir(argv[1]) == -1) {
        fprintf(stderr, "opendir, chdir error for %s\n", argv[1]);
        exit(1);
    }
    while((dentry = readdir(dirp)) != NULL) {
        if([          ] == 0)
            continue;
        memcpy([          ]);
        if(stat([          ]) == -1) {
            fprintf(stderr, "stat error for %s\n", filename);
            break;
        }
        if((statbuf.st_mode & S_IFMT) == S_IFREG)
            printf("%-14s %ld\n", [          ]);
        else
            printf("%-14s\n", [          ]);
    }
    exit(0);
}
```

실행결과

```
//ssu_oslab.c 파일은 파일크기를 위한 더미파일임
root@localhost:/home/oslab# mkdir ssu_oslab
root@localhost:/home/oslab# cd ssu_oslab
```

```
root@localhost:/home/oslab/ssu_oslab# vi ssu_oslab.c
root@localhost:/home/oslab/ssu_oslab# gcc -o ssu_oslab_exec ssu_oslab.c
root@localhost:/home/oslab/ssu_oslab# mkdir ssu_dir
root@localhost:/home/oslab/ssu_oslab# cd ..
root@localhost:/home/oslab# ./ssu_directory ssu_oslab
..
.
ssu_dir
ssu_oslab.c     102
ssu_oslab_exec 8656
```

27. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1.5점]

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

#define PATH_MAX     1024

int main(void)
{
    char *pathname;

    if (chdir("/home/oslab") < 0) {
        fprintf(stderr, "chdir error\n");
        exit(1);
    }

    pathname = [                              ];
    if (getcwd([                              ]) == NULL) {
        fprintf(stderr, "getcwd error\n");
        exit(1);
    }

    printf("current directory = %s\n", [                  ]);
    exit(0);
}
```

실행결과

```
oslab@localhost:~/ssu_osdir$ ./ssu_getcwd
current directory = /home/oslab
```

28. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2점]

<ssu_test.txt>

```
Linux System Programming!
Unix System Programming!
Linux Mania
Unix Mania
```

<ssu_fopen.c>

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

```

```
int main(void)
{
    char *fname = "ssu_test.txt";
    char *mode = [          ];

    if (fopen([                    ]) == NULL) {
        fprintf(stderr, "fopen error for %s\n",[          ]);
        exit(1);
    }
    else
        printf("Success!\nFilename: <%s>, mode: <%s>\n",[                    ]);

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_fopen
Success!
Filename: <ssu_test.txt>, mode: <r>
```

29. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2점]

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

int main(void)
{
    char *fname = "ssu_test.txt";
    [                ] ;

    if ((fp = fopen([                ])) == NULL) {
        fprintf(stderr, "fopen error for %s\n",[                ]);
        exit(1);
    }
    else {
        printf("Success!\n");
        printf("Opening \"%s\" in \"r\" mode!\n",[                ]);
    }

    fclose(fp);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_fclose
Success!
Opening "ssu_test.txt" in "r" mode!
```

30. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1.5점]

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

int main(void)
{
    int character;

    while ((character = [                              ])
        if (putc([                    ]) == EOF) {
            fprintf(stderr, "standard output error\n");
            exit(1);
        }

    if (ferror([          ])) {
        fprintf(stderr, "standard input error\n");
        exit(1);
    }

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_getc
Hello stdin, stdout
Hello stdin, stdout
```

31. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1.5점]

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

#define BUFFER_SIZE 1024

int main(void)
{
    char buf[BUFFER_SIZE];

    while (fgets([                                ]) != NULL)
        if (fputs([                  ]) == EOF) {
            fprintf(stderr, "standard output error\n");
            exit(1);
        }

    if (ferror([          ])) {
        fprintf(stderr, "standard input error\n");
        exit(1);
    }

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_fgets
Hi fgets, fputs
Hi fgets, fputs
```

32. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [3.5점]

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

int main(void)
{
    char operator;
    [                ];
    int character;
    int number = 0;
    // fopen()은 읽기 전용으로 open
    if ((fp = fopen([                ])) == NULL) {
        fprintf(stderr, "fopen error for ssu_expr.txt\n");
        exit(1);
    }

    while ([          ])) {
        while ((character = [                ])
            number = [                ] - 48;
        fprintf(stdout, " %d\n", number);
        number = 0;

        if (character != EOF) {
            [                ];
            operator = [          ];
            printf("Operator => %c\n", operator);
        }
    }

    fclose(fp);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# vi ssu_expr.txt
root@localhost:/home/oslab# ./ssu_ungetc
 123
Operator => +
 456
Operator => *
 789
Operator =>
 0
```

<ssu_expr.txt>

```
123+456*789
```

33. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 잘못된 부분을 고치시오. [2.5점]

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

#define BUFFER_SIZE 1024

int main (int argc, char *argv[])
{
    char buf[BUFFER_SIZE];
    FILE  *fp;

    if (argc != 2) {
        fprintf(stderr, "usage: %s <file>\n", argv[0]);
        exit(1);
    }

    if ((fp = fopen(argv[1], "w")) == NULL) {
        fprintf(stderr, "fopen error for %s\n", argv[0]);
        exit(1);
    }

    fputs("Input String >> ", stdin);
    gets(buf);
    fputs(buf);
    rewind(fp);
    fgets(buf, fp);
    puts(buf);
    fclose(fp);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_fputs ssu_hello.txt
Input String >> Hello OSLAB~!
Hello OSLAB~!
root@localhost:/home/oslab# cat ssu_hello.txt
Hello OSLAB~!root@localhost:/home/oslab#
```

34. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1점]

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

struct ssu_id {
    char name[64];
    int id;
};

int main(void)
{
    struct ssu_id test1, test2;
    char *fname = "ssu_exam.dat";
```

```
    FILE *fp;

    if ((fp = fopen(fname, "w")) == NULL) {
        fprintf(stderr, "fopen error for %s\n", fname);
        exit(1);
    }

    printf("Input ID>> ");
    scanf("%d", &test1.id);
    printf("Input name>> ");
    scanf("%s", test1.name);

    if (fwrite(                                          ) != 1) {
        fprintf(stderr, "fwrite error\n");
        exit(1);
    }

    fclose(fp);

    if ((fp = fopen(fname, "r")) == NULL) {
        fprintf(stderr, "fopen error for %s\n", fname);
        exit(1);
    }

    if (fread(                                           ) != 1) {
        fprintf(stderr, "fread error\n");
        exit(1);
    }

    printf("\nID     name\n");
    printf(" ============\n");
    printf("%d       %s\n", test2.id, test2.name);
    fclose(fp);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_fwrite
Input ID>> 23
Input name>> OSLAB

ID     name
 ============
23       OSLAB
```

35. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 잘못된 부분을 고치시오. [2점]

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

int main(void)
{
```

```
    char *fname = "ssu_test.txt";
    FILE *fp;
    long position;
    int character;

    if ((fp = fopen(fname, "w")) == NULL) {
        fprintf(stderr, "fopen error for %s\n", fname);
        exit(1);
    }

    printf("Input number >>");
    scanf("%ld", position);
    lseek(fp, position, SEEK_END);
    character = getc(fp);
    printf("%ldth character => %c\n", position, character);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_fseek
Input number >>8
8th character => y
```

36. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1점]

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

int main(void)
{
    char *fname = "ssu_test.txt";
    FILE *fp;
    long fsize;

    if ((fp = fopen(fname, "r")) == NULL) {
        fprintf(stderr, "fopen error for %s\n", fname);
        exit(1);
    }

    fseek(                              );
    fsize =                             ;
    printf("The size of <%s> is %ld bytes\n", fname, fsize);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ls -al ssu_test.txt
-rw-rw-r-- 1 oslab oslab 74 Jan  9 18:48 ssu_test.txt
root@localhost:/home/oslab# ./ssu_ftell
The size of <ssu_test.txt> is 74 bytes
```

37. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1.5점]

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

```
#include <stdlib.h>

#define BUFFER_SIZE 1024

int main(void)
{
    char *fname = [          ];
    char name[BUFFER_SIZE];
    FILE* fp;
    int age;

    fp = fopen(fname, "r");
    fscanf([                    ]);
    fclose(fp);
    fp = fopen([                ]);
    fprintf(fp, "%s is %d years old\n", name, age);
    fclose(fp);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# cat > ssu_test.dat
HongGilDong 20
root@localhost:/home/oslab# ./ssu_fscanf
root@localhost:/home/oslab# cat ssu_test.dat
HongGilDong is 20 years old
```

38. 프로그램 실행 시 아래 실행결과의 빈 칸을 채우시오. [2점]

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

char buf1[] = "abcdefghij";
char buf2[] = "ABCDEFGHIJ";

#define FILE_MODE (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

int main(void)
{
    int fd;

    if ((fd = creat("ssu_file.hole", FILE_MODE)) < 0) {
        fprintf(stderr, "creat error");
        exit(1);
    }

    if (write(fd, buf1, 10) != 10) {
        fprintf(stderr, "buf1 write error");
        exit(1);
    }
```

```
    if (lseek(fd, 16384, SEEK_SET) == -1) {
        fprintf(stderr, "lseek error");
        exit(1);
    }

    if (write(fd, buf2, 10) != 10) {
        fprintf(stderr, "buf2 write error");
        exit(1);
    }

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./hole
root@localhost:/home/oslab# ls -l ssu_file.hole
[              ] 1 root root [      ] 4월 27 20:04 ssu_file.hole
root@localhost:/home/oslab# od -c ssu_file.hole
[                                                                    ]
```

39. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [3.5점]

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

#define BUFFER_SIZE 1024

int main(void)
{
    char buf[BUFFER_SIZE];
    char *fname = "ssu_test.txt";
    int count;
    int fd1, fd2;

    fd1 = open(fname, O_RDONLY, 0644);
    fd2 = open(fname, O_RDONLY, 0644);

    if ([                                ]) {
        fprintf(stderr, "open error for %s\n", fname);
        exit(1);
    }

    count = read([                    ], 25);
    buf[count] = 0;
    printf("fd1's first printf : %s\n", buf);
```

```
    lseek(                              );
    count = read(                       , 24);
    buf[count] = 0;
    printf("fd1's second printf : %s\n", buf);
    count = read(                       , 25);
    buf[count] = 0;
    printf("fd2's first printf : %s\n", buf);
    lseek(                              );
    count = read(                       , 24);
    buf[count] = 0;
    printf("fd2's second printf : %s\n", buf);

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_read_1
fd1's first printf :Linux System Programming!
fd1's second printf :Unix System Programming!
fd2's first printf :Linux System Programming!
fd2's second printf :Unix System Programming!
```

40. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [4점]

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

#define TABLE_SIZE 128
#define BUFFER_SIZE 1024

int main(int argc, char *argv[])
{
    static struct {
          long offset;
          int length;
    } table [TABLE_SIZE];
    char buf[BUFFER_SIZE];
    long offset;
    int entry;
    int i;
    int length;
    int fd;
        if (argc < 2) {
            fprintf(stderr, "usage: %s <file>\n", argv[0]);
            exit(1);
    }
    if ((fd = open(argv[1], O_RDONLY)) < 0) {
        fprintf(stderr, "open error for %s\n", argv[1]);
        exit(1);
    }
```

```
    entry = 0;
    offset = 0;
    while ((length = read(                                  )) > 0) {
        for (i = 0 ; i < length ; i++) {
            ;
            ;
            if (buf[i] == '\n')
                table[++entry].offset = offset;
        }
    }
#ifdef DEBUG
    for (i = 0 ; i < entry ; i++)
        printf("%d : %ld, %d\n", i + 1, table[i].offset, table[i].length);
#endif
    while (1) {
        printf("Enter line number : ");
        scanf(                          );

        if (                          )
            break;

        (fd, table[length].offset, 0);
        if (read(                                       )<= 0)
            continue;

                                        = '\0';
        printf("%s", buf);
    }

    close(fd);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_read_3 /etc/group
Enter line number : 1
root:x:0:
Enter line number : 2
daemon:x:1:
Enter line number : 3
bin:x:2:
Enter line number : 4
sys:x:3:
Enter line number : 1
root:x:0:
Enter line number : -1
```

41. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [2점]

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

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

#define S_MODE 0644
#define BUFFER_SIZE 1024

int main(int argc, char *argv[])
{
    char buf[BUFFER_SIZE];
    int fd1, fd2;
    int length;

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

    if ((fd1 = open(                              )) < 0) {
        fprintf(stderr, "open error for %s\n", argv[1]);
        exit(1);
    }

    if ((fd2 = open(                                                        )) < 0) {
        fprintf(stderr, "open error for %s\n", argv[2]);
        exit(1);
    }

    while ((length = read(                                  )) > 0)
        write(                          );

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_write_1 /etc/passwd password
root@localhost:/home/oslab# ls -l /etc/passwd password
-rw-r--r-- 1 root root 2240 Jan  2 23:55 /etc/passwd
-rw-r--r-- 1 root root 2240 Jan  3 05:27 password
```

42. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1.5점]

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

#define RWRWRW (                                                        )

int main(void)
{
    ;
```

```
    if (creat("ssu_test1", RWRWRW) < 0) {
        fprintf(stderr, "creat error for ssu_test1\n");
        exit(1);
    }

    umask(                                                    );

    if (creat("ssu_test2", RWRWRW) < 0) {
        fprintf(stderr, "creat error for ssu_test2\n");
        exit(1);
    }

    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# umask
0022
root@localhost:/home/oslab# ./ssu_umask
root@localhost:/home/oslab# ls -l ssu_test1 ssu_test2
-rw-rw-rw- 1 root root 0  1월 11 21:20 ssu_test1
-rw------- 1 root root 0  1월 11 21:20 ssu_test2
root@localhost:/home/oslab# umask
0022
```

43. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [1.5점]

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

#define MODE_EXEC (S_IXUSR|S_IXGRP|S_IXOTH)

int main(int argc, char *argv[])
{
    struct stat statbuf;
    int i;

    if (argc < 2) {
        fprintf(stderr, "usage: %s <file1> <file2> ... <fileN>\n", argv[0]);
        exit(1);
    }
    for (i = 1; i < argc; i++) {
        if (stat(argv[i], &statbuf) < 0) {
            fprintf(stderr, "%s : stat error\n", argv[i]);
            continue;
        }
        ;
        ;
        if (chmod(                    ) < 0)
            fprintf(stderr, "%s : chmod error\n", argv[i]);
```

```
        else
            printf("%s : file permission was changed.\n", argv[i]);
    }
    exit(0);
}
```

실행결과 [일반 사용자 권한으로 실행]

```
oslab@localhost:~$ mkdir ssu_test_dir
oslab@localhost:~$ ./ssu_chmod_1 /bin/su /home/oslab/ssu_test_dir
/bin/su : chmod error
/home/oslab/ssu_test_dir : file permission was changed.
```

44. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 빈 칸을 채우시오. [7.5점]

<ssu_osdir/ssu_dir1/ssu_file1>

```
Hi,
it's the Keyword
Bye.
```

<ssu_osdir/ssu_dir2/ssu_file2>

```
Hi,
it's not the keyword.
Bye.
```

<ssu_directory_1.c>

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

#ifdef  PATH_MAX
static int pathmax = PATH_MAX;
#else
static int pathmax = 0;
#endif
#define MAX_PATH_GUESSED 1024

#ifndef LINE_MAX
#define LINE_MAX 2048
#endif

char pathname;
char command[LINE_MAX], grep_cmd[LINE_MAX];
int ssu_do_grep(void) {
    struct dirent *dirp;
    struct stat statbuf;




    if (lstat(pathname, statbuf) < 0) {
        fprintf(stderr, "lstat error for %s\n", pathname);
```

```
        return 0;
    }

    if (S_ISDIR(statbuf.stat_mode) == 0) {
        sprintf(command, "%s %s", grep_cmd, pathname);
        printf("%s : \n", pathname);
        system(command);
        return 0;
    }
    ptr = pathname + sizeof(pathname);
    *ptr++ = '\0';
    *ptr = '/';
    if ((dp = opendir(pathname)) == NULL) {
        fprintf(stderr, "opendir error for %s\n", pathname);
        return 0;
    }
    while ((dirp = read(dp)) == NULL)
        if (strcmp(dirp->d_name, ".") || strcmp(dirp->d_name, "..")) {
            strcpy(ptr, dirp->dir_name);
            if (ssu_do_grep() < 0)
                break;
        }

    ptr[-1] = 0;
    closedir(dp);
    return 0;
}
void ssu_make_grep(int argc, char *argv[]) {
    int i;
    strcpy(grep_cmd, " grep");
    for (i = 1;i < argc-1;i++) {
        strcat(grep_cmd, argv[i]);
    }
}

int main(int argc, char *argv[])
{
    if (argc < 2) {
        fprintf(stderr, "usage: %s <-CVbchilnsvwx> <-num> <-A num> <-B num> <-f file> \n"
               "             <-e> expr <directory>\n", argv[0]);
        exit(1);
    }

    if (pathmax == 0) {
        if ((pathmax = pathconf("/", _PC_PATH_MAX)) < 0)
            pathmax = MAX_PATH_GUESSED;
        else
            pathmax++;
    }
```

```
    if ((pathname = (pathmax+1)) == NULL) {
        fprintf(stderr, "malloc error\n");
        exit(1);
    }
    strcpy(pathname, argv[2]);
    ssu_make_grep(argc, argv);
    ssu_do_grep();
    exit(0);
}
```

실행결과

```
oslab@localhost:~$ vi ssu_osdir/ssu_dir1/ssu_file1
oslab@localhost:~$ vi ssu_osdir/ssu_dir2/ssu_file2
oslab@localhost:~$ ./ssu_directory_1 -n Keyword /home/oslab/ssu_osdir
/home/oslab/ssu_osdir/ssu_dir2/ssu_file2 :
/home/oslab/ssu_osdir/ssu_dir1/ssu_file1 :
2:it's the Keyword
```

45. 프로그램 실행 시 아래 실행결과가 나올 수 있도록, 잘못된 부분을 고치시오. [2점]

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

int main(void)
{
    char buf[64];
    char fname = "ssu_tempfile";
    int fd;
    int length;

    if ((fd = open(fname, O_RDWR | O_TRUNC, 0600)) < 0) {
        fprintf(stderr, "first open error for %s\n", fname);
        exit(1);
    }

    if (link(fname) < 0) {
        fprintf(stderr, "unlink error for %s\n", fname);
        exit(1);
    }

    if (write(fd, "How are you?", 12) != 12) {
        fprintf(stderr, "write error\n");
        exit(1);
    }

    lseek(fd, 0, SEEK_CUR);

    if ((length = read(fd, buf, sizeof(buf))) < 0) {
        fprintf(stderr, "buf read error\n");
        exit(1);
```

```
    }
    else
        buf[length] = 0;

    printf("%s\n", buf);
    close(fd);
    if ((fd = open(fname, O_RDWR)) < 0) {
        fprintf(stderr, "second open error for %s\n", fname);
        exit(1);
    }
    else
        close(fd);
    exit(0);
}
```

실행결과

```
root@localhost:/home/oslab# ./ssu_unlink_1
How are you?
second open error for ssu_tempfile
```

※ 다음은 ssu_shell의 일부분이다. 물음에 답하시오. [46~47]

46. oldtime과 newtime변수를 이용해 소요시간이 출력되도록 빈 칸을 채우시오 [3.5점]

```
#define BUFFER_SIZE

void get_command(void)
{
    int argc;
    struct timeval oldtime, newtime;
    char command[10][BUFFER_SIZE];
    char input_command[BUFFER_SIZE];

    fgets(input_command,BUFFER_SIZE,stdin);
    *(input_command+mystrlen(input_command)-1) = '\0';
    argc = mytok(input_command,' ',command);

    gettimeofday(                                );
    if (mystrcmp(command[0],PROGRAM) != 0)
        system(input_command);
    else if(init_option(argc,command) < 0)
        replace_all();
    gettimeofday(                                );

    if(oldtime.tv_usec > newtime.tv_usec) {
        newtime.tv_usec =                                    ;
        newtime.tv_sec =                                    ;
    }
    else {
        newtime.tv_usec =                                    ;
        newtime.tv_sec =                                    ;
    }
```

```
    newtime.tv_usec = [                                    ];
    printf("\ntime : %ld.%03ld\n",newtime.tv_sec, newtime.tv_usec);
}
```

47. 다음은 ssu_sed에서 inquriy_dir 함수를 재귀호출하여 디렉터리를 탐색하는 프로그램이다. depth가 maxdepth를 초과하거나 더 이상 하위 디렉토리가 없으면 재귀호출이 멈출 수 있도록 빈 칸을 채우시오 [1.5점]

```
#define BUFFER_SIZE 1024

void inquriy_dir(const char * mPath, int depth) {
    DIR *dir;
    struct dirent *entry;
    struct stat statbuf;

    if(dFlag && depth > maxdepth) return;

    if ((dir = opendir(mPath)) == NULL) {
        fputs("Error for open directory ",stdout);
        puts(mPath);
        return;
    }

    if ((entry = readdir(dir)) == NULL) {
        fputs("Error for read directory ",stdout);
        puts(mPath);
        return;
    }

    do {
        char path[BUFFER_SIZE];
        int len;

        strcpy(path, mPath);
        strcat(path, "/");
        strcat(path, entry->d_name);
        len = strlen(mPath) + strlen(entry->d_name) + 1;
        path[len] = 0;

        lstat(path,&statbuf);
        if (entry->d_type == DT_DIR) {
            if (strcmp(                              ) == 0 || strcmp(                              ) == 0)
                continue;

            inquriy_dir(                              );
        }
        else if (S_IXUSR & statbuf.st_mode) {
            fputs(root_path_full,stdout);
            fputs(path,stdout);
            fputs(" : ",stdout);
            fputs("cannot rewrite executable file\n", stdout);
        }
```

```
        else if (entry->d_type == DT_REG)
            replace_file(path);
    } while ((entry = readdir(dir)) != NULL);

    closedir(dir);
}
```

※ 다음 물음에 답하시오. [48~49]

48. 아래 gcc의 실행결과를 확인할 수 있는 옵션을 빈 칸에 채우시오. (단, -Wall 이 아닌 구체적인 옵션을 사용) [1점]

<gcc_option1.c>

```
#include <stdio.h>

int foo();

int main(void)
{
    foo();
    return 0;
}

int foo()
{
    printf("foo\n");
}
```

실행결과

```
root@localhost:/home/oslab# gcc gcc_option1.c
root@localhost:/home/oslab# gcc [                              ] gcc_option1.c
gcc_option1.c: In function 'foo':
gcc_option1.c:13.1: warning: control reaches end of non-void function [                              ]
 }
 ^
root@localhost:/home/oslab#
```

49. 아래 gcc의 실행결과를 확인할 수 있는 옵션을 빈 칸에 채우시오. (단, -Wall 이 아닌 구체적인 옵션을 사용) [1점]

<gcc_option3.c>

```
#include <stdio.h>

int main(void)
{
    char arr[] = "Hello World\n";
    for(int i = 0; i < sizeof(arr); i++)
        printf("%c", arr[i]);
    return 0;
}
```

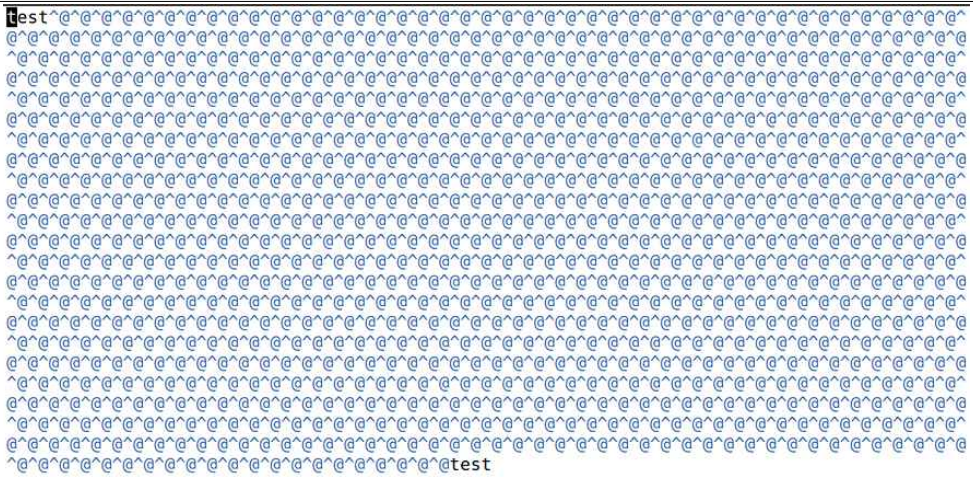
실행결과

```
root@localhost:/home/oslab# gcc gcc_option3.c
root@localhost:/home/oslab# gcc [                              ] gcc_option3.c
gcc_option3.c: In function 'main':
gcc_option3.c:6.20: warning: comparison between signed and unsigned integer expressions[                    ]
 for(int i = 0; i < sizeof(arr); i++)
```

```
                    ^
root@localhost:/home/oslab#
```

(보너스 문제) 다음 프로그램은 인자로 주어진 (1) 한 개의 파일(a.txt)을 오픈하여, (2) 파일의 크기를 확인하고, (3) 파일 오프셋을 처음으로 이동하고, (4) 파일을 모두 read하여 buf 버퍼에 저장하고, (5) 파일을 닫는다. (6) 파일의 크기를 0으로 초기화하여 오픈하고, (7) 원래 a.txt 파일에서 홀(\0)을 만나기 전까지의 내용을 (6)에서 오픈한 파일에 write하고, (8) 홀(\0)이후 파일의 마지막 오프셋까지 (6)에서 오픈한 파일에 write를 하고, (9) 파일을 닫는 프로그램을 작성하시오. (단, creat()는 사용하지 못하며 단, 에러 처리만을 위해서 fprintf() 사용 가능함. (1), (2), (3), (4), (6), (7), (8)은 반드시 기본적인 에러 처리가 포함되어야 됨) [10점 - 컴파일 시 error 발생시 0점 처리, warning 1개당 1점 감점]

<a.txt>



<ssu_hole.c>

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

int main(int argc, char *argv[])
{
        int fd;
        int size;
        char *buf;
        int i;
```

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

실행결과

```
ssu-oslab@localhost:~/lsp$ od -c a.txt
0000000   t   e   s   t  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0002000   t   e   s   t  \n
0002005
ssu-oslab@localhost:~/lsp$ ./a.out a.txt
ssu-oslab@localhost:~/lsp$ od -c a.txt
0000000   t   e   s   t   t   e   s   t  \n
0000011
ssu-oslab@localhost:~/lsp$
```