System Programming & OS 실습
3. File I/O

최민국, 정지헌, 안석현, 김선재
Dankook University
{mgchoi, wlgjsjames7224, seokhyun, rlatjswo0824}@dankook.ac.kr

Index

❖ open(), Read()

❖ write()

❖ mycat

❖ create new file

❖ lseek

System call

❖ Allowing a process to request a kernel service.

❖ The primary interface between processes and the operating system, providing a means to invoke services made available by the operating system [Operating System Concepts 10th]

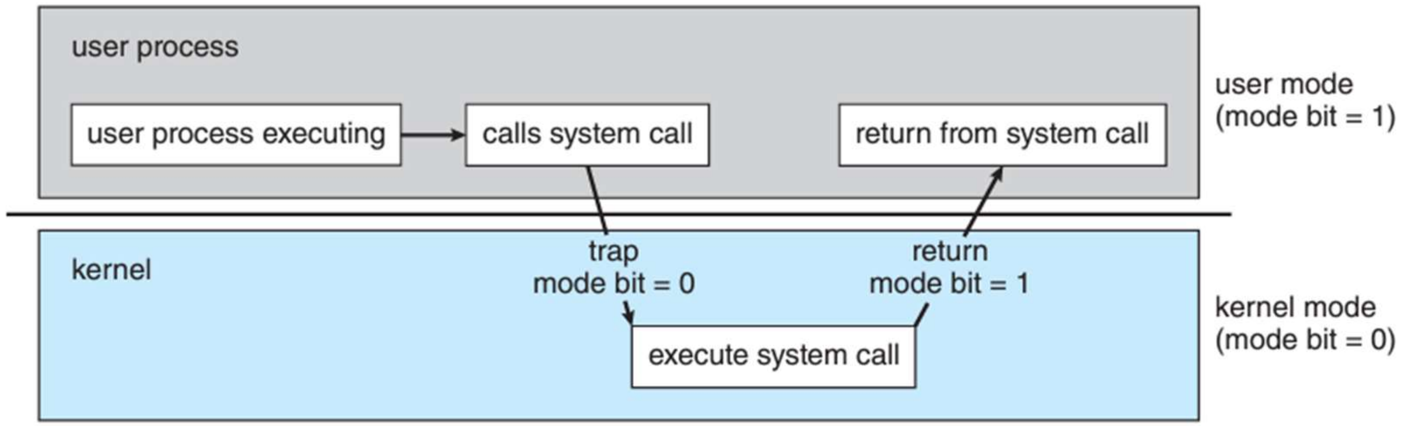


Figure 1.13 Transition from user to kernel mode.

System call

❖ Work

- File I/O, Process management, network, memory…

[Operating System Concepts 10th]

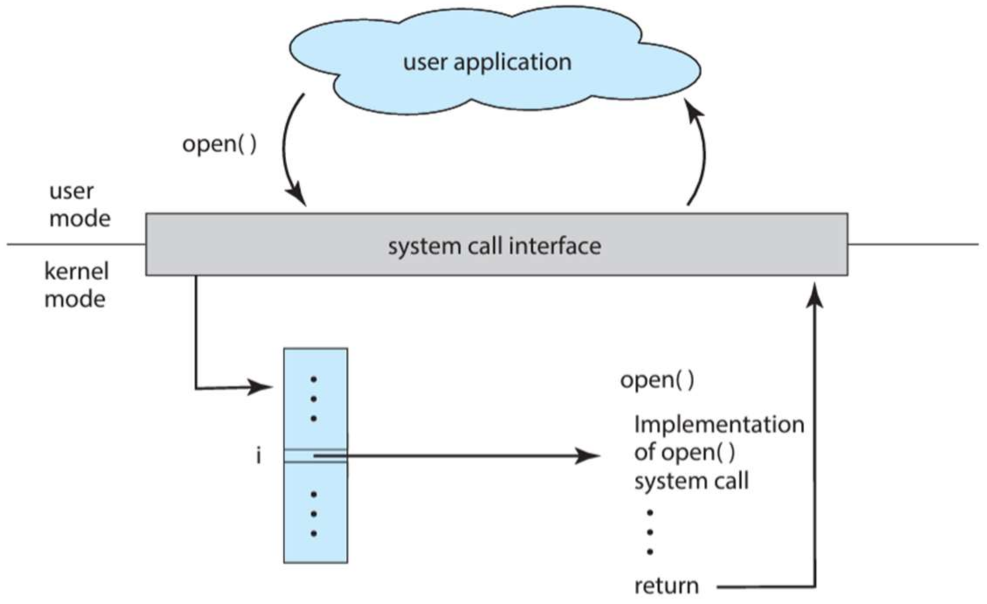


Figure 2.6 The handling of a user application invoking the `open()` system call.

open(), read()

❖ open()

- parameters(const char *pathname, int flags, mode_t mode)
  - *const char pathname: The path of the file to be opened
  - Int flags: specifies the access mode of the file

    (e.g., O_RDONLY, O_WRONLY, O_RDWR, O_CREAT: 파일 생성, O_EXCL: 파일이 존재할 시 -1 반환)
  - O_CREAT 일 시, 파라미터 mode_t mode 호출
- return value
  - On a successful read, the number of non-negative integer.
  - If an error occurs, read() return -1.

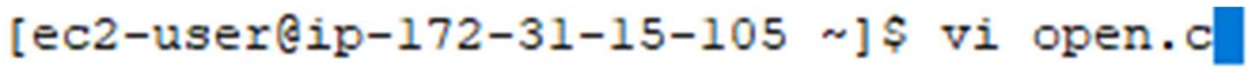
https://man7.org/linux/man-pages/index.html

open(), read()

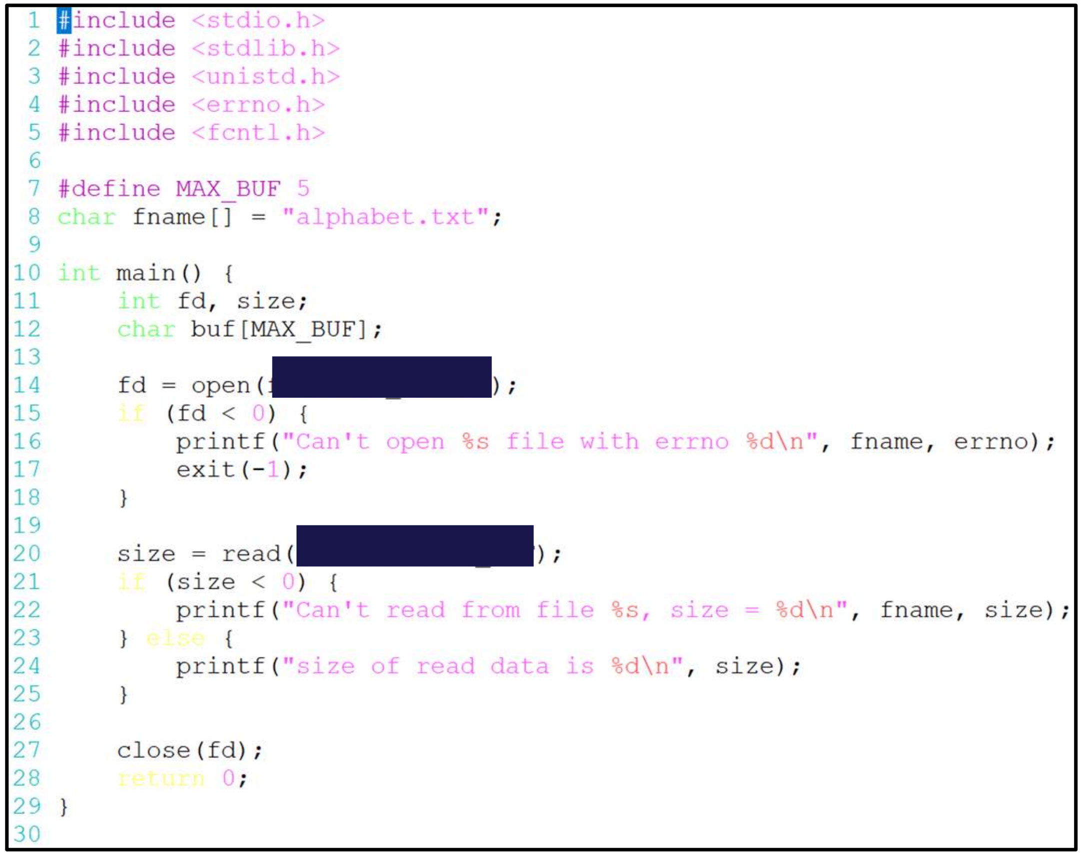
- read()
  - parameters(int fd, void *buf, size_t count)
    - int fd: the file descriptor to be read
    - void *buf: a buffer into which the data will be read
    - size_t count: the maximum number of bytes to be read into the buffer
  - return value
    - On a successful read, the number of bytes read is returned
    - A return value of 0 indicates end of file
    - If an error occurs, read() return -1.

https://man7.org/linux/man-pages/index.html

open(), read()

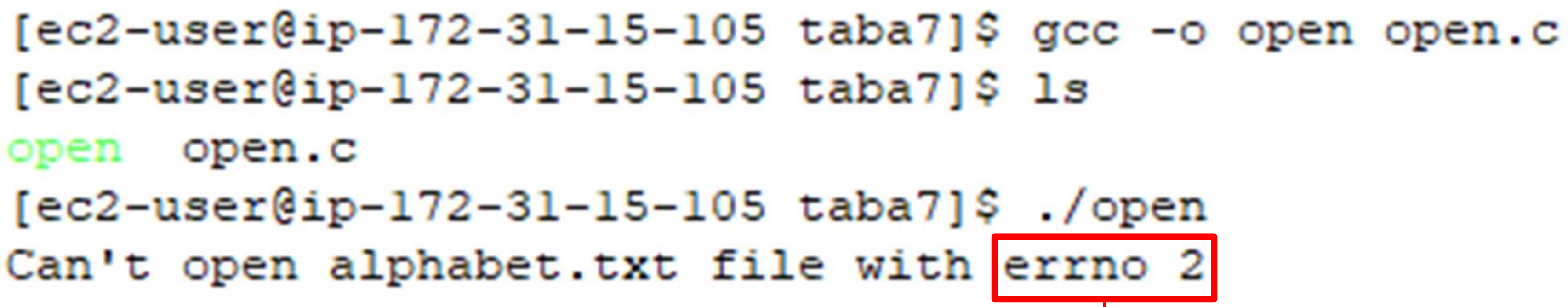


```
[ec2-user@ip-172-31-15-105 ~]$ vi open.c
```



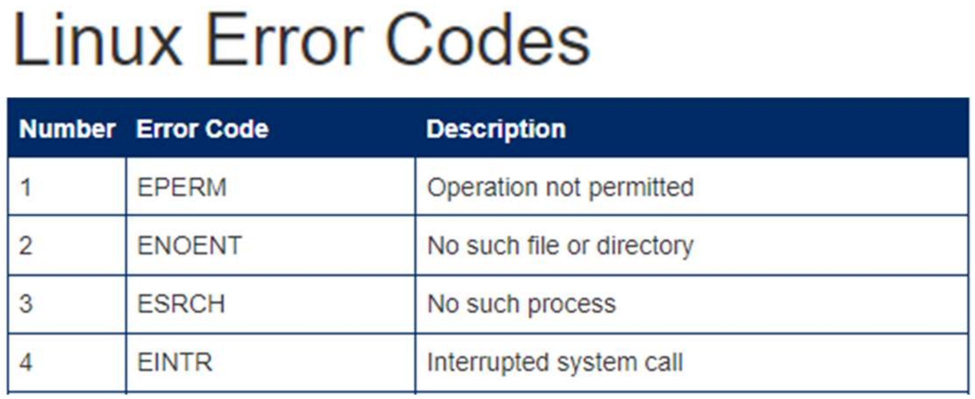
```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>

#define MAX_BUF 5
char fname[] = "alphabet.txt";

int main() {
    int fd, size;
    char buf[MAX_BUF];

    fd = open([redacted]);
    if (fd < 0) {
        printf("Can't open %s file with errno %d\n", fname, errno);
        exit(-1);
    }

    size = read([redacted]);
    if (size < 0) {
        printf("Can't read from file %s, size = %d\n", fname, size);
    } else {
        printf("size of read data is %d\n", size);
    }

    close(fd);
    return 0;
}

```

open(), read()



```
[ec2-user@ip-172-31-15-105 taba7]$ gcc -o open open.c
[ec2-user@ip-172-31-15-105 taba7]$ ls
open  open.c
[ec2-user@ip-172-31-15-105 taba7]$ ./open
Can't open alphabet.txt file with errno 2
```

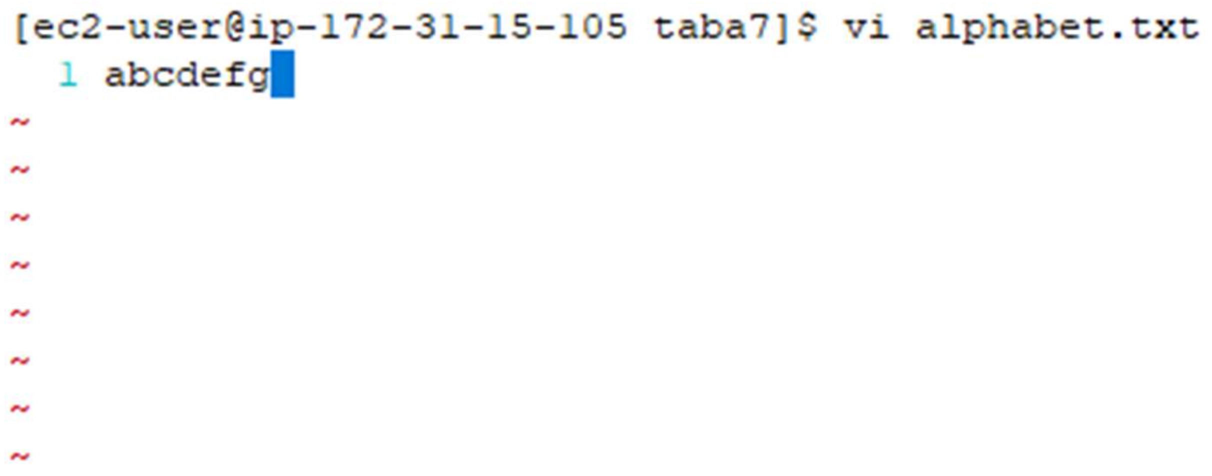
파일 및 디렉토리 X



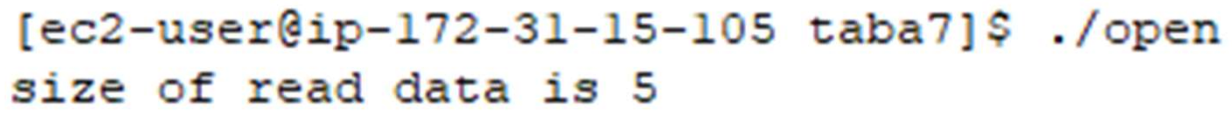
Linux Error Codes

Number	Error Code	Description
1	EPERM	Operation not permitted
2	ENOENT	No such file or directory
3	ESRCH	No such process
4	EINTR	Interrupted system call

open(), read()



```
[ec2-user@ip-172-31-15-105 taba7]$ vi alphabet.txt
  abcdefg
~
~
~
~
~
~
~
~
```



```
[ec2-user@ip-172-31-15-105 taba7]$ ./open
size of read data is 5
```

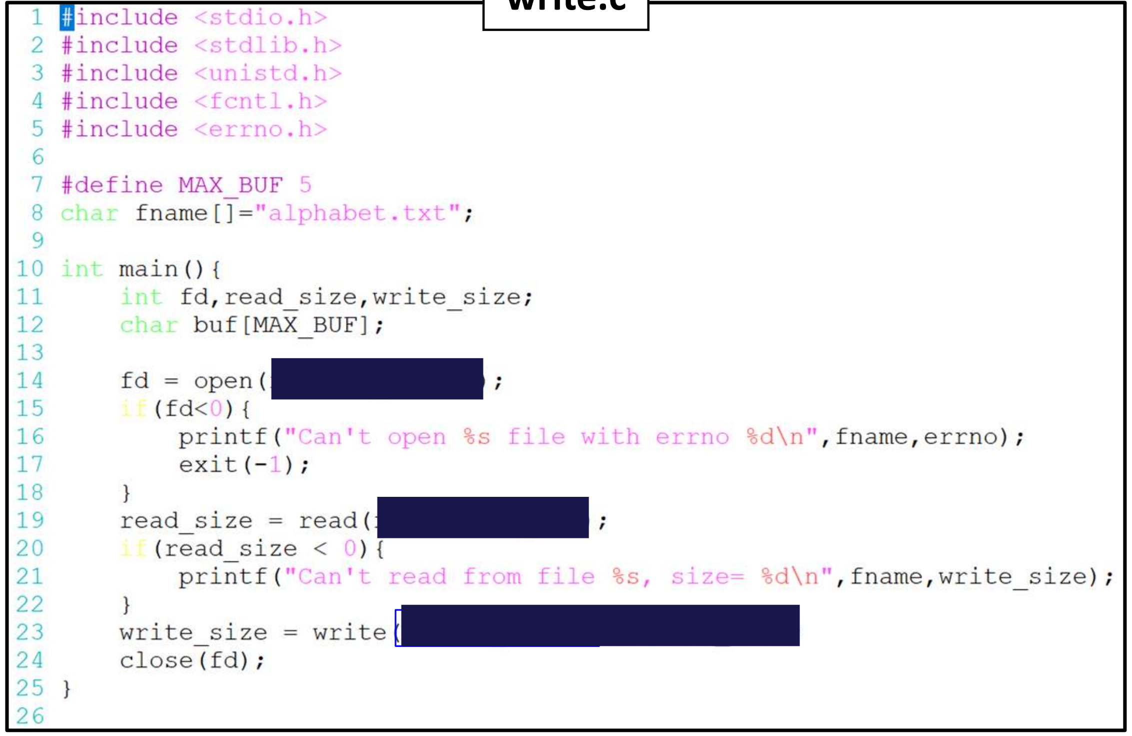
Write()

- write()
  - parameters(int fd, const void *buf, size_t count)
    - int fd: the file descriptor to be write
    - void *buf: a buffer into which the data will be write
    - size_t count: the maximum number of bytes to be write into the buffer
  - return value
    - On a successful read, the number of bytes write is returned
    - If an error occurs, read() return -1.

https://man7.org/linux/man-pages/index.html

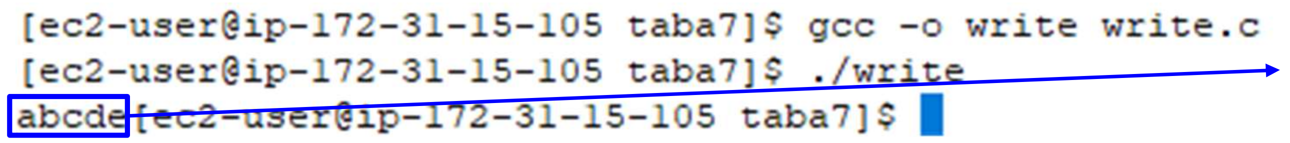
Write()

write.c



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

#define MAX_BUF 5
char fname[]="alphabet.txt";

int main(){
    int fd,read_size,write_size;
    char buf[MAX_BUF];

    fd = open([redacted];
    if(fd<0){
        printf("Can't open %s file with errno %d\n",fname,errno);
        exit(-1);
    }
    read_size = read([redacted];
    if(read_size < 0){
        printf("Can't read from file %s, size= %d\n",fname,write_size);
    }
    write_size = write([redacted]
    close(fd);
}

```

STDIN_FILENO:표준 입력
STOUT_FILENO: 표준 출력



```
[ec2-user@ip-172-31-15-105 taba7]$ gcc -o write write.c
[ec2-user@ip-172-31-15-105 taba7]$ ./write
abcde[ec2-user@ip-172-31-15-105 taba7]$ 
```

5개만 읽고 출력
Why?
두 가지 방법

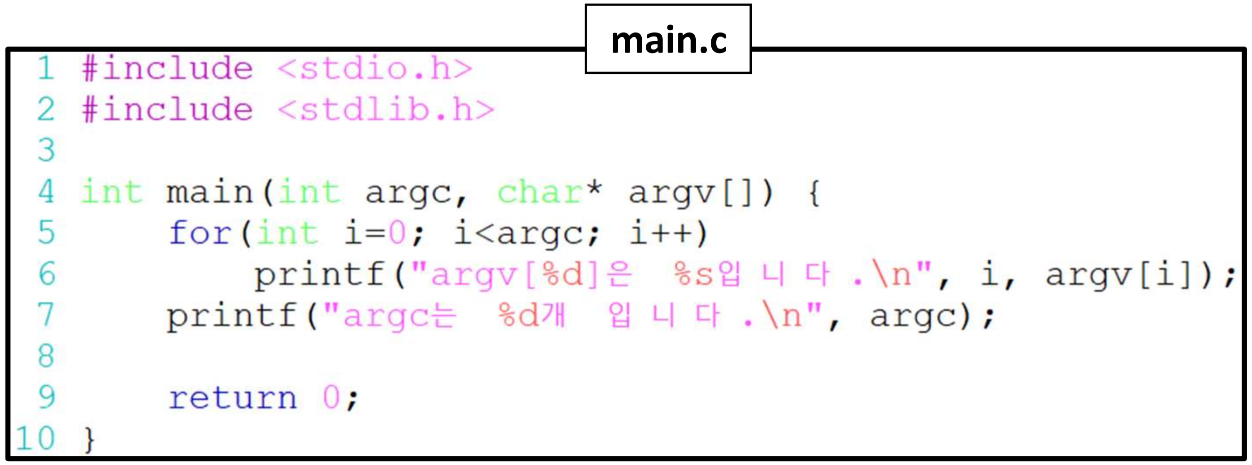
Write_1.c
Write_2.c

main()

❖ main(int argc, char* argv)

- int argc: main함수에 전달되는 인자의 개수 + 1
- char* argv[0]: 실행된 프로그램의 경로와 프로그램 이름
- char* argv[1]: 첫번째 인자
- . 두번째 인자
- . 세번째 인자

main.c



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

int main(int argc, char* argv[]) {
    for(int i=0; i<argc; i++)
        printf("argv[%d]은 %s입니다.\n", i, argv[i]);
    printf("argc는 %d개 입니다.\n", argc);

    return 0;
}
```

❖ 실습) main 프로그램 실행

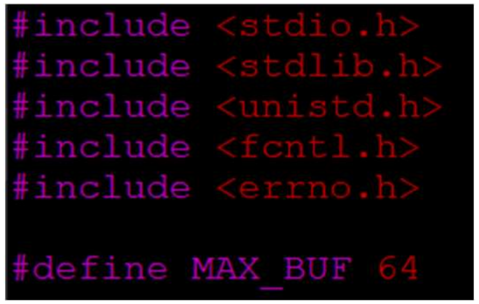
⇒ 인자 5개 전달 후 결과 확인

⇒ Input: *./main 1 2 3 4 5* or *[main path]/main 1 2 3 4 5*

mycat-1

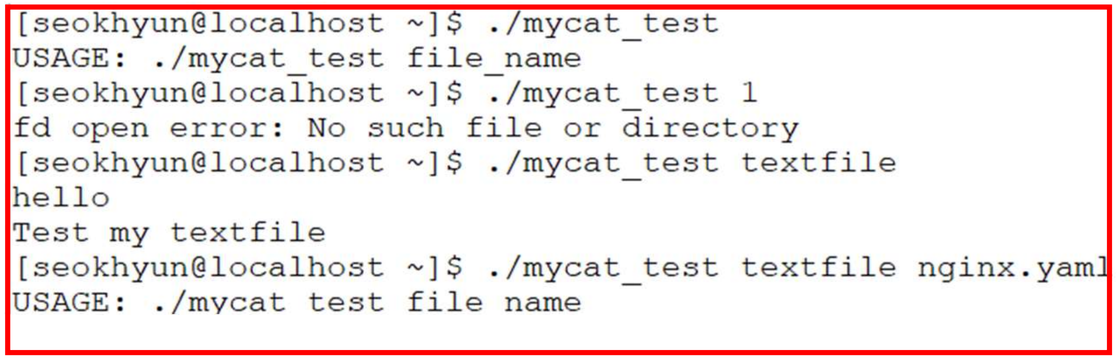
mycat.c

```
int main(int argc, char *argv[]) {
    // 변수 선언
    int fd, read_size, write_size;
    char buf[MAX_BUF];

    // 인수 개수 확인 (예외 처리)
    if [redacted]) {
        printf("USAGE: %s file_name\n", argv[0]);
        exit(-1);
    }

    // 파일 열기
    fd = [redacted];
    if (fd < 0) {
        perror("fd open error");
        exit(-1);
    }

    // 파일 읽기 및 출력
    while ((read_size = [redacted]) > 0) {
        write_size = [redacted]);
        if (write_size != read_size) {
            perror("write error");
            close(fd);
            exit(-1);
        }
    }

    if (read_size < 0) {
        perror("read error");
        close(fd);
        exit(-1);
    }

    // 파일 닫기
    if (close(fd) < 0) {
        perror("close error");
        exit(-1);
    }

    return 0;
```



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

#define MAX_BUF 64
```

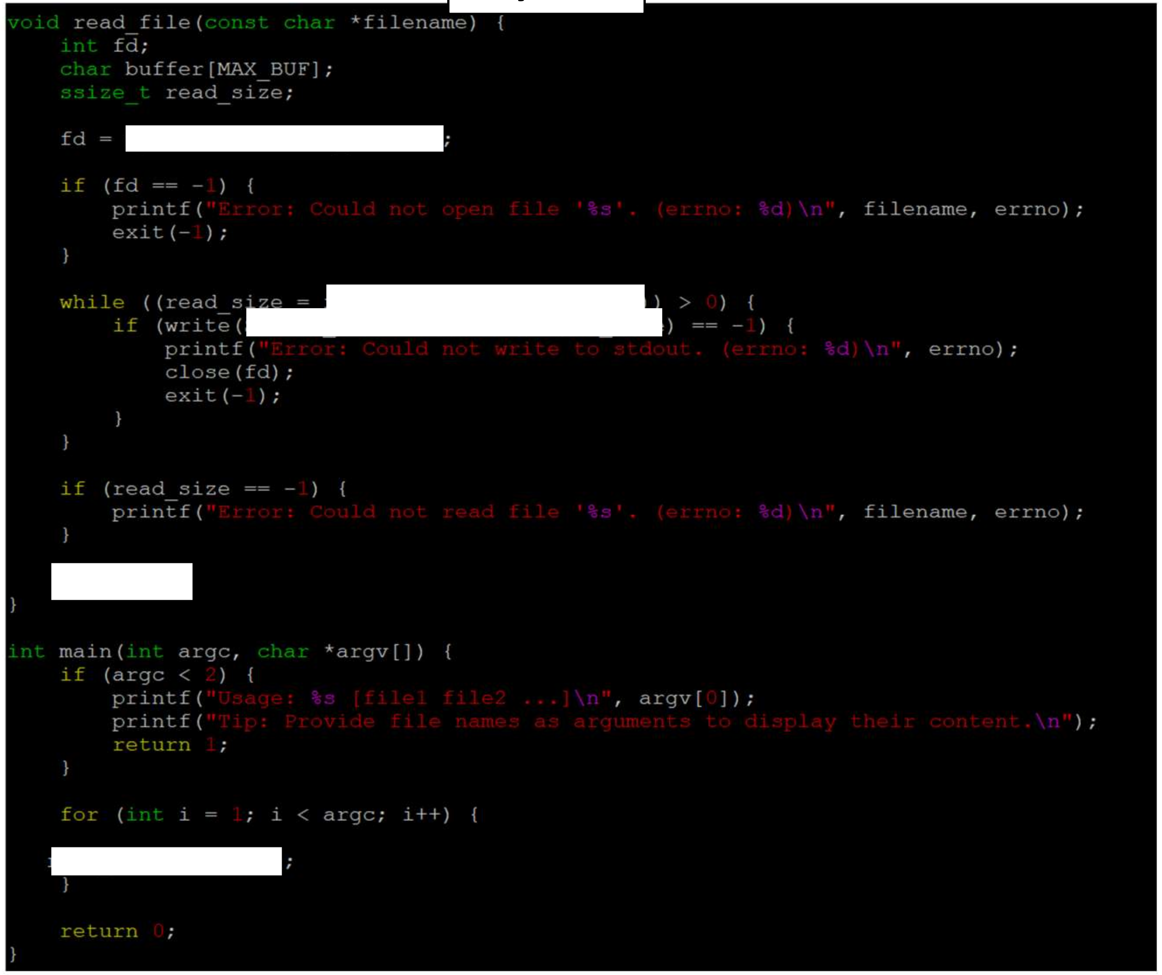
Hint: Output



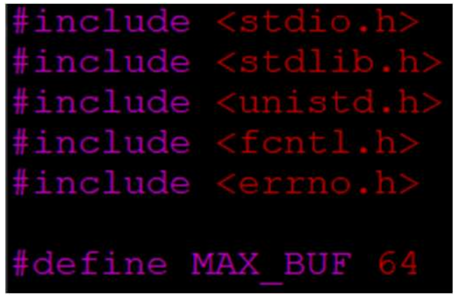
```
[seokhyun@localhost ~]$ ./mycat_test
USAGE: ./mycat_test file_name
[seokhyun@localhost ~]$ ./mycat_test 1
fd open error: No such file or directory
[seokhyun@localhost ~]$ ./mycat_test textfile
hello
Test my textfile
[seokhyun@localhost ~]$ ./mycat_test textfile nginx.yaml
USAGE: ./mycat_test file_name
```

mycat-2

mycat.c

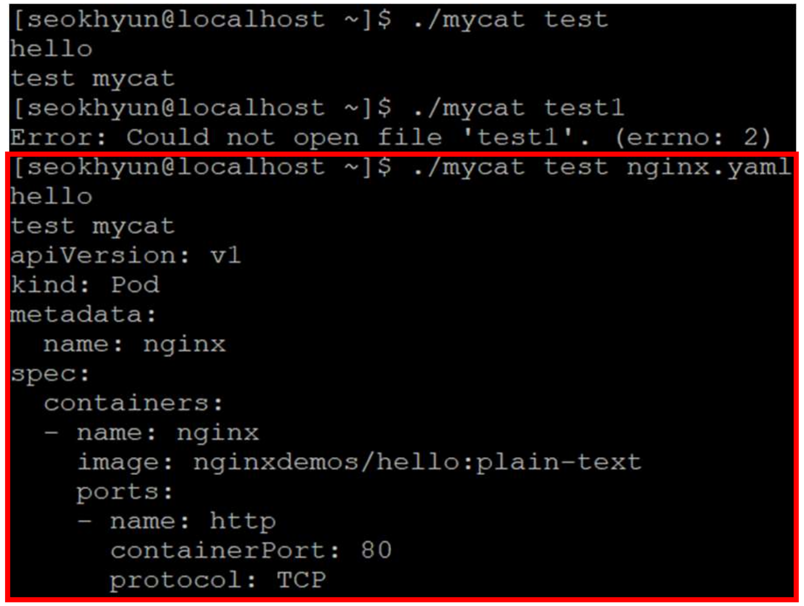


```c
void read_file(const char *filename) {
    int fd;
    char buffer[MAX_BUF];
    ssize_t read_size;

    fd =                               ;

    if (fd == -1) {
        printf("Error: Could not open file '%s'. (errno: %d)\n", filename, errno);
        exit(-1);
    }

    while ((read_size =                               ) > 0) {
        if (write(                                   ) == -1) {
            printf("Error: Could not write to stdout. (errno: %d)\n", errno);
            close(fd);
            exit(-1);
        }
    }

    if (read_size == -1) {
        printf("Error: Could not read file '%s'. (errno: %d)\n", filename, errno);
    }

    
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s [file1 file2 ...]\n", argv[0]);
        printf("Tip: Provide file names as arguments to display their content.\n");
        return 1;
    }

    for (int i = 1; i < argc; i++) {
                              ;
    }

    return 0;
}
```



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

#define MAX_BUF 64
```

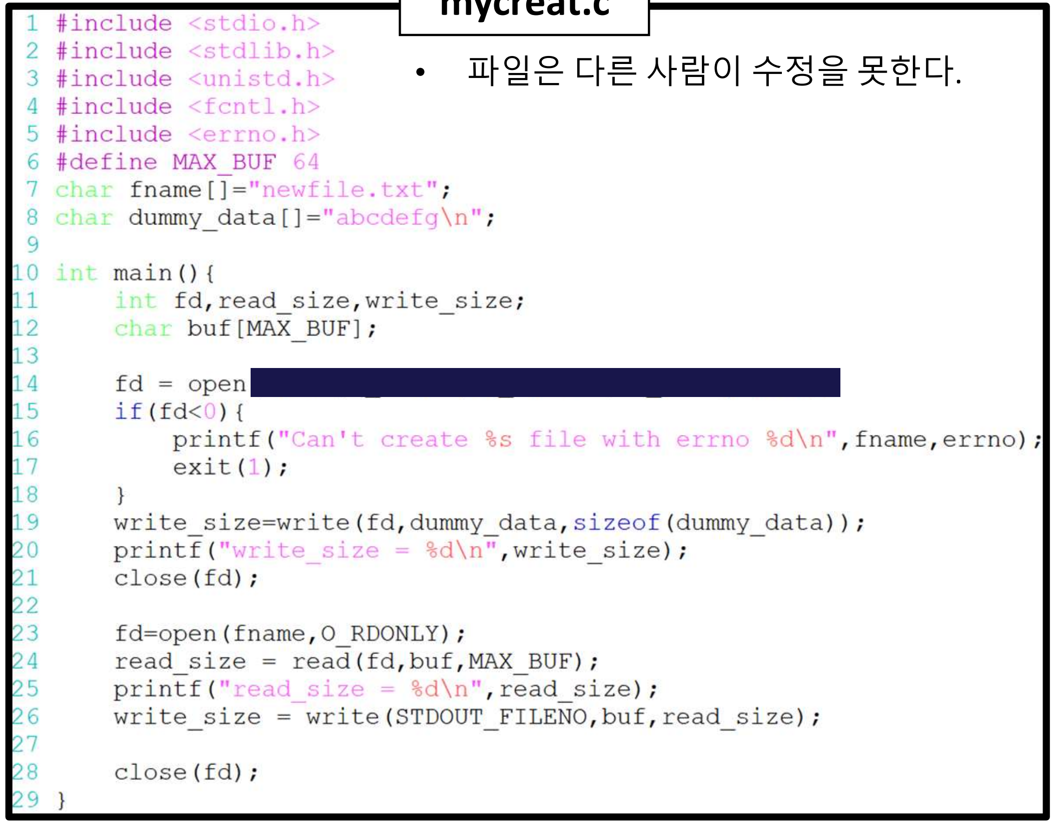
Hint: Output



```
[seokhyun@localhost ~]$ ./mycat test
hello
test mycat
[seokhyun@localhost ~]$ ./mycat test1
Error: Could not open file 'test1'. (errno: 2)
[seokhyun@localhost ~]$ ./mycat test nginx.yaml
hello
test mycat
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginxdemos/hello:plain-text
    ports:
    - name: http
      containerPort: 80
      protocol: TCP
```

Create new file

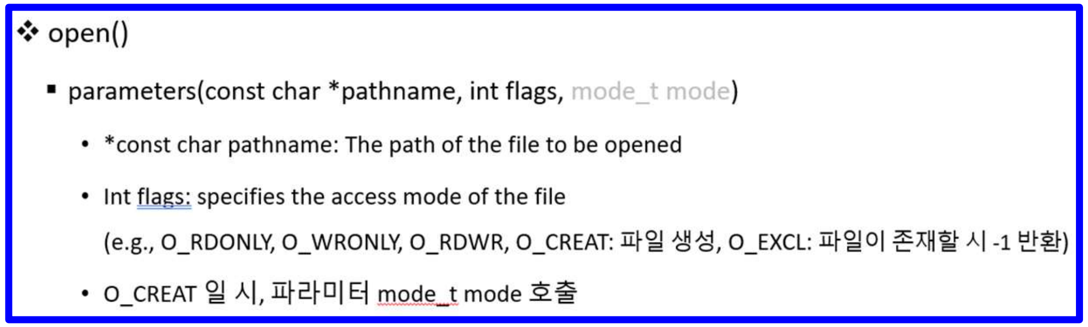
mycreat.c



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
char fname[]="newfile.txt";
char dummy_data[]="abcdefg\n";

int main(){
    int fd,read_size,write_size;
    char buf[MAX_BUF];

    fd = open[redacted]
    if(fd<0){
        printf("Can't create %s file with errno %d\n",fname,errno);
        exit(1);
    }
    write_size=write(fd,dummy_data,sizeof(dummy_data));
    printf("write_size = %d\n",write_size);
    close(fd);

    fd=open(fname,O_RDONLY);
    read_size = read(fd,buf,MAX_BUF);
    printf("read_size = %d\n",read_size);
    write_size = write(STDOUT_FILENO,buf,read_size);

    close(fd);
}
```

- 파일은 다른 사람이 수정을 못한다.

Hint 1: parameter

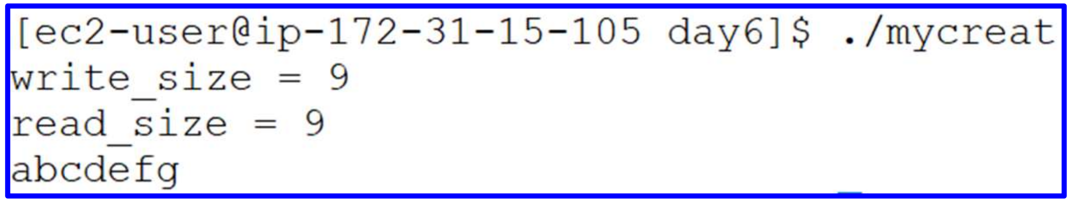


❖ open()

- parameters(const char *pathname, int flags, mode_t mode)
  - *const char pathname: The path of the file to be opened
  - Int flags: specifies the access mode of the file
    (e.g., O_RDONLY, O_WRONLY, O_RDWR, O_CREAT: 파일 생성, O_EXCL: 파일이 존재할 시 -1 반환)
  - O_CREAT 일 시, 파라미터 mode_t mode 호출

Hint 2:
mode_t mode 0664

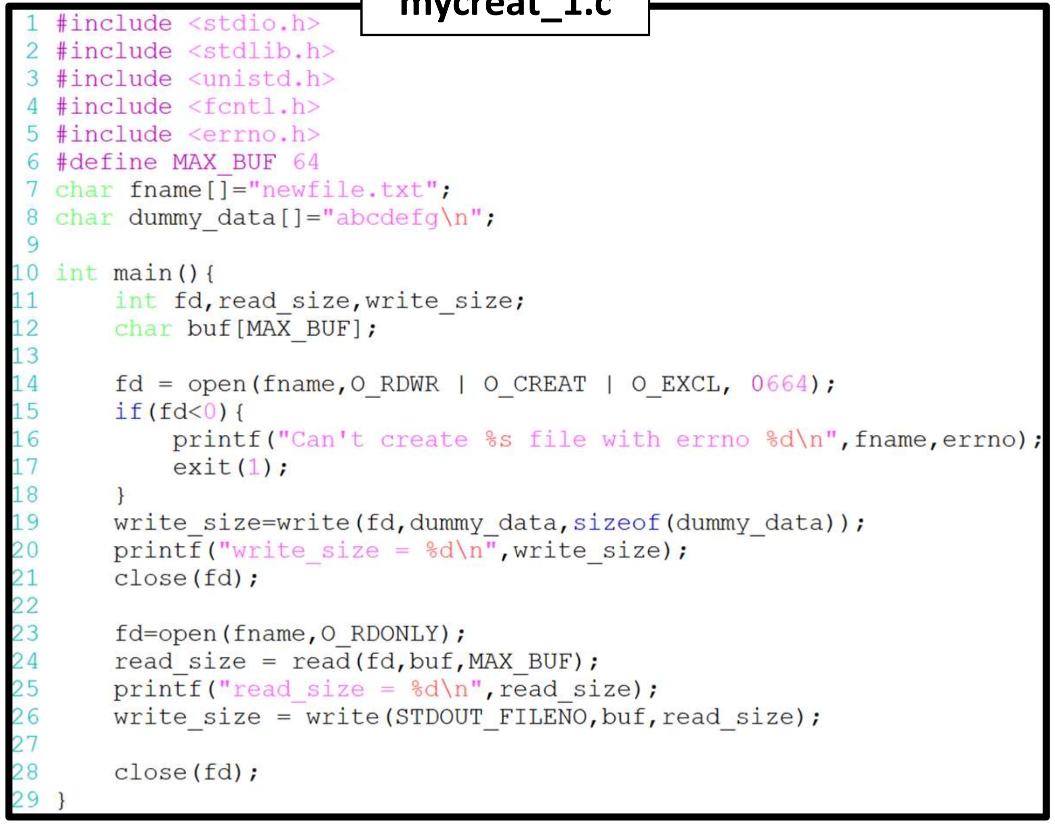
Hint 3: Output



```
[ec2-user@ip-172-31-15-105 day6]$ ./mycreat
write_size = 9
read_size = 9
abcdefg
```

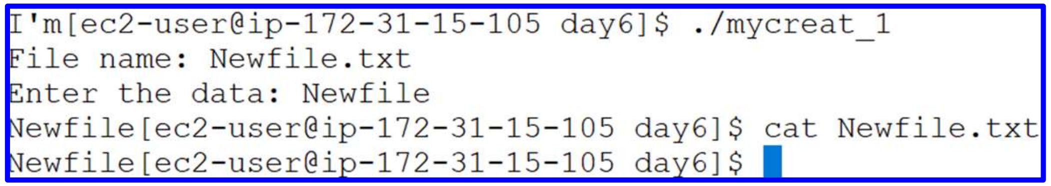
Create new file

mycreat_1.c



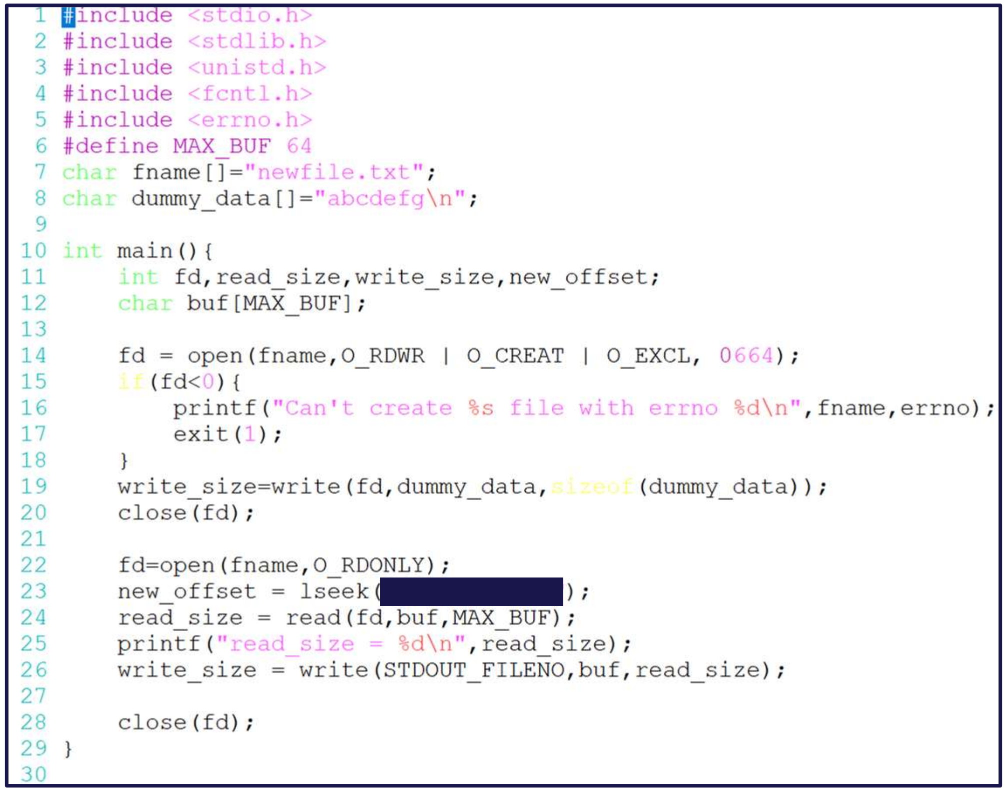
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
char fname[]="newfile.txt";
char dummy_data[]="abcdefg\n";

int main(){
    int fd,read_size,write_size;
    char buf[MAX_BUF];

    fd = open(fname,O_RDWR | O_CREAT | O_EXCL, 0664);
    if(fd<0){
        printf("Can't create %s file with errno %d\n",fname,errno);
        exit(1);
    }
    write_size=write(fd,dummy_data,sizeof(dummy_data));
    printf("write_size = %d\n",write_size);
    close(fd);

    fd=open(fname,O_RDONLY);
    read_size = read(fd,buf,MAX_BUF);
    printf("read_size = %d\n",read_size);
    write_size = write(STDOUT_FILENO,buf,read_size);

    close(fd);
}
```

Hint: Output



```
I'm[ec2-user@ip-172-31-15-105 day6]$ ./mycreat_1
File name: Newfile.txt
Enter the data: Newfile
Newfile[ec2-user@ip-172-31-15-105 day6]$ cat Newfile.txt
Newfile[ec2-user@ip-172-31-15-105 day6]$
```

lseek()

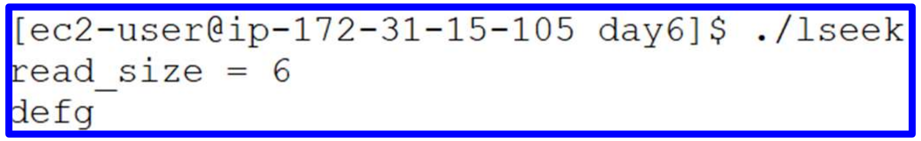


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
char fname[]="newfile.txt";
char dummy_data[]="abcdefg\n";

int main(){
    int fd,read_size,write_size,new_offset;
    char buf[MAX_BUF];

    fd = open(fname,O_RDWR | O_CREAT | O_EXCL, 0664);
    if(fd<0){
        printf("Can't create %s file with errno %d\n",fname,errno);
        exit(1);
    }
    write_size=write(fd,dummy_data,sizeof(dummy_data));
    close(fd);

    fd=open(fname,O_RDONLY);
    new_offset = lseek([redacted]);
    read_size = read(fd,buf,MAX_BUF);
    printf("read_size = %d\n",read_size);
    write_size = write(STDOUT_FILENO,buf,read_size);

    close(fd);
}

```

Hint: Parameter

https://man7.org/linux/man-pages/man2/lseek.2.html

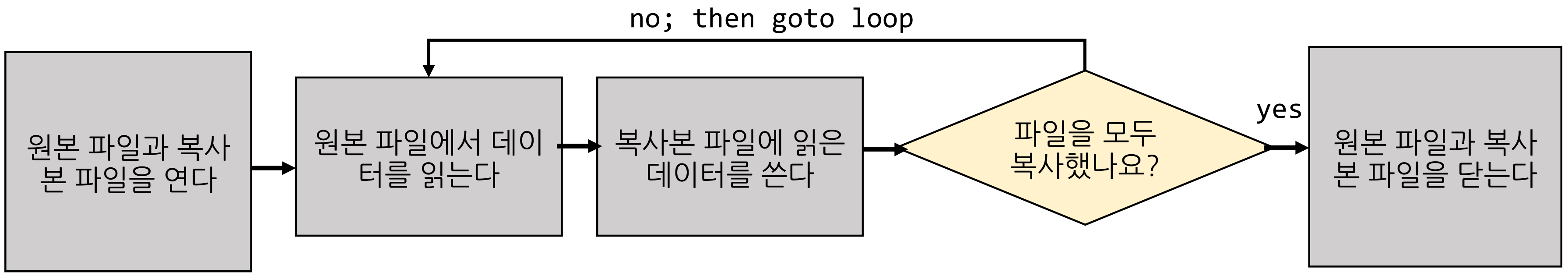
Hint: Output



```
[ec2-user@ip-172-31-15-105 day6]$ ./lseek
read_size = 6
defg
```

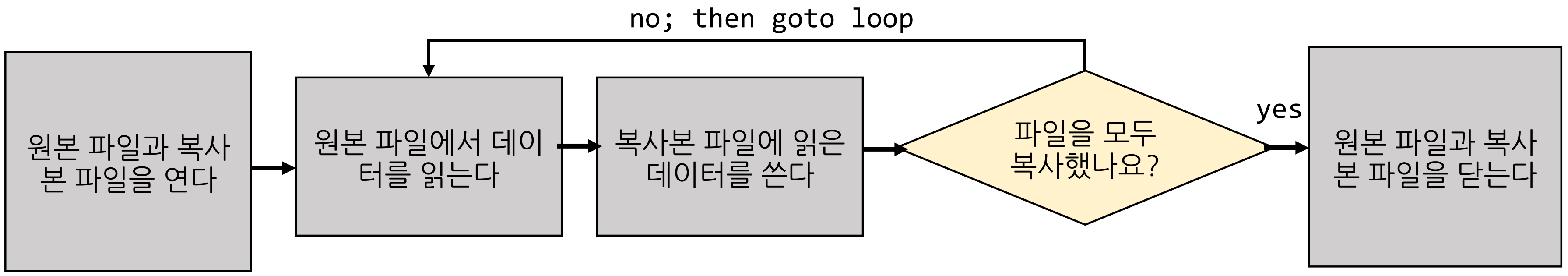
mycp

- 프로그램 요구사항 기술 - mycp
  - Input
    - USAGE : `./mycp origin_file_here dest_file_here`
    - 내용이 적혀있는 원본파일origin file
  - Output
    - 원본 파일origin file 의 user data가 적혀있는 복사본 파일destination file
  - mycp algorithm


no; then goto loop
원본 파일과 복사본 파일을 연다
원본 파일에서 데이터를 읽는다
복사본 파일에 읽은 데이터를 쓴다
파일을 모두 복사했나요?
yes
원본 파일과 복사본 파일을 닫는다

mycp

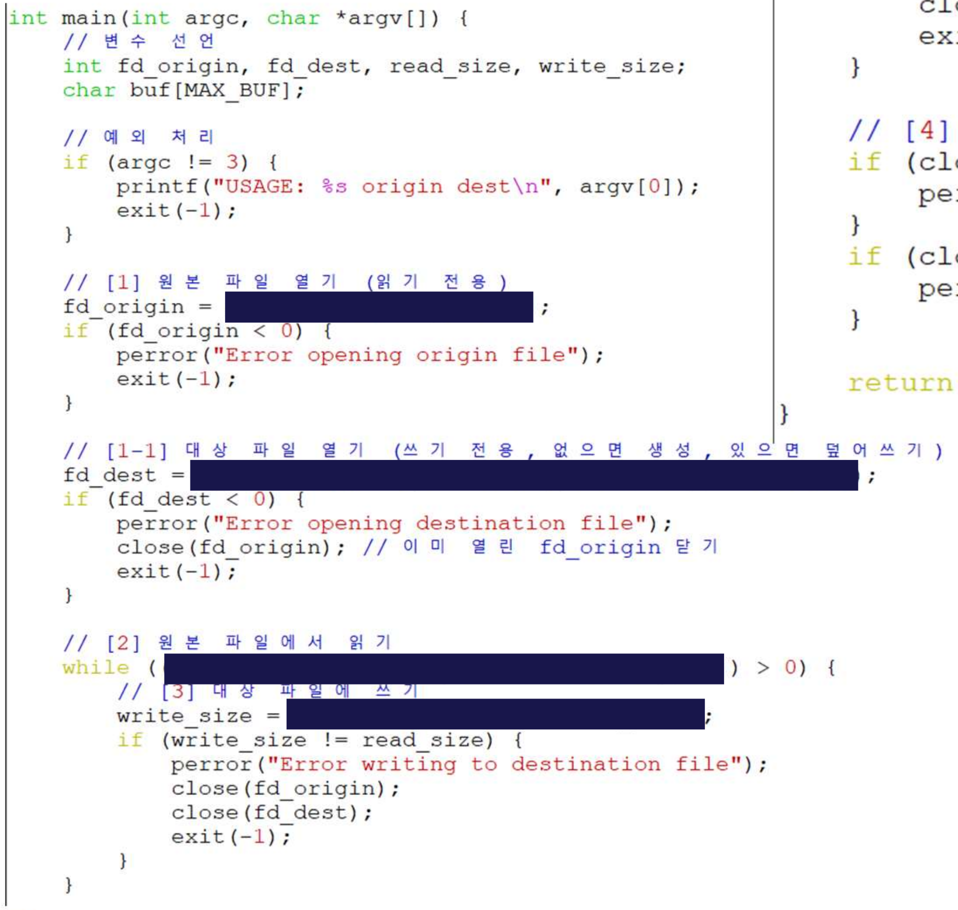
- 프로그램 요구사항 기술 - mycp
  - Input
    - USAGE : `./mycp origin_file_here dest_file_here`
    - 내용이 적혀있는 원본파일origin file
  - Output
    - 원본 파일origin file 의 user data가 적혀있는 복사본 파일destination file
  - mycp algorithm


no; then goto loop
원본 파일과 복사본 파일을 연다
원본 파일에서 데이터를 읽는다
복사본 파일에 읽은 데이터를 쓴다
파일을 모두 복사했나요?
yes
원본 파일과 복사본 파일을 닫는다

mycp

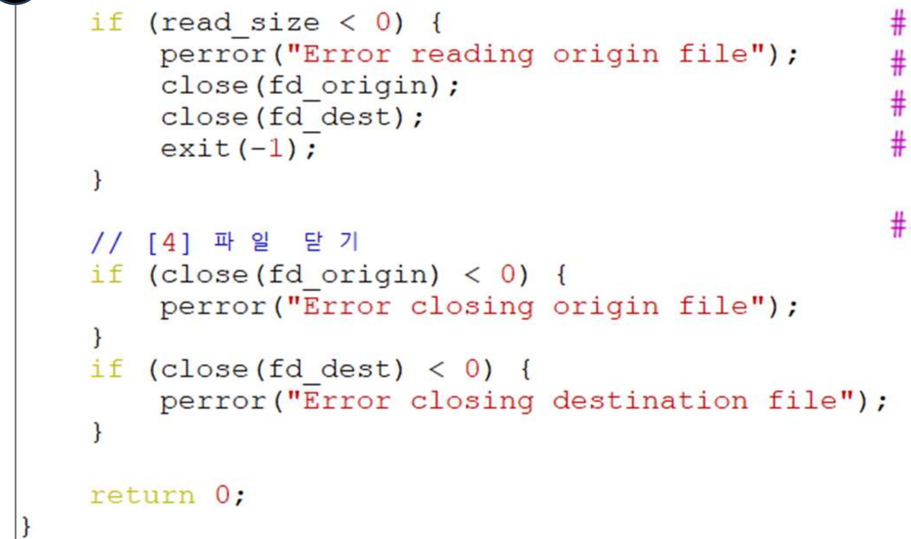
mycp.c

1

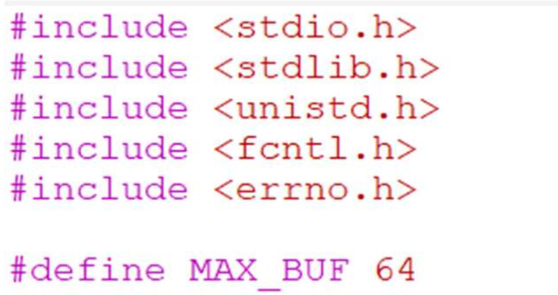


```c
int main(int argc, char *argv[]) {
    // 변수 선언
    int fd_origin, fd_dest, read_size, write_size;
    char buf[MAX_BUF];

    // 예외 처리
    if (argc != 3) {
        printf("USAGE: %s origin dest\n", argv[0]);
        exit(-1);
    }

    // [1] 원본 파일 열기 (읽기 전용)
    fd_origin = [redacted];
    if (fd_origin < 0) {
        perror("Error opening origin file");
        exit(-1);
    }

    // [1-1] 대상 파일 열기 (쓰기 전용, 없으면 생성, 있으면 덮어쓰기)
    fd_dest = [redacted];
    if (fd_dest < 0) {
        perror("Error opening destination file");
        close(fd_origin); // 이미 열린 fd_origin 닫기
        exit(-1);
    }

    // [2] 원본 파일에서 읽기
    while ([redacted]) > 0) {
        // [3] 대상 파일에 쓰기
        write_size = [redacted];
        if (write_size != read_size) {
            perror("Error writing to destination file");
            close(fd_origin);
            close(fd_dest);
            exit(-1);
        }
    }
```

2

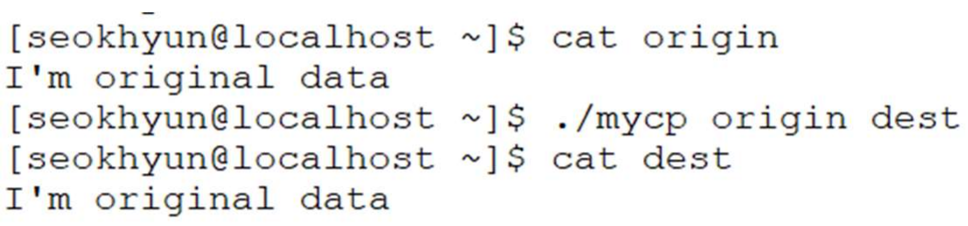


```c
    if (read_size < 0) {
        perror("Error reading origin file");
        close(fd_origin);
        close(fd_dest);
        exit(-1);
    }

    // [4] 파일 닫기
    if (close(fd_origin) < 0) {
        perror("Error closing origin file");
    }
    if (close(fd_dest) < 0) {
        perror("Error closing destination file");
    }

    return 0;
}
```



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

#define MAX_BUF 64
```

Hint: Output



```
[seokhyun@localhost ~]$ cat origin
I'm original data
[seokhyun@localhost ~]$ ./mycp origin dest
[seokhyun@localhost ~]$ cat dest
I'm original data
```

그 이외 시스템 콜

- create()
- mkdir(), readdir(), rmdir()
- pipe()
- mknod()
- link(),unlink()
- dup(),dup2()
- stat(),fstat()
- chmod(), fchmod()
- Ioctl(), fcntl()
- Sync(), fsync()