8장. 파일 처리

표준 입출력 함수

ANSI C 표준 입출력 함수

◈ 정수형 파일 기술자 대신 **FILE** 구조를 이용

기능	함수 원형
파일 열기/닫기	FILE *fopen(const char *path, const char *mode); int fclose(FILE *stream);

파일 읽기/쓰기	int fgetc(FILE *stream);
	int fputc(int c, FILE *stream);
	char *fgets(char *s, int size, FILE *stream);
	int fputs(const char *s, FILE *stream);
	int fscanf(FILE *stream, const char *format, …);
	int fprintf(FILE *stream, const char *format, …);

파일 위치 재배치	int fseek(FILE *stream, long offset, int whence);
	long ftell(FILE *stream);
	void rewind(FILE *stream);

표준 입출력 함수

▣ 표준 입출력 함수 사용 예

◈ 텍스트 파일 내용 복사 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_copy.c
 */

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

main(int argc, char *argv[])
{
    FILE *src;  /* source file      */
    FILE *dst;  /* destination file */
    char ch;
    int count = 0;

    if (argc < 3) {
                printf("Usage:  file_copy  source_file
destination_file₩n");
        exit(1);
    }

    if ( (src = fopen(argv[1], "r")) == NULL ) {
        perror("fopen: src");
        exit(1);
    }

    if ( (dst = fopen(argv[2], "w")) == NULL ) {
        perror("fopen: dst");
        exit(1);
    }

    while ( !feof(src) ) {
        ch = (char) fgetc(src);
        if ( ch != EOF )
            fputc((int)ch, dst);
        count++;
    }

    fclose(src);
    fclose(dst);
}
```

표준 입출력 함수

▣ 표준 입출력 함수 사용 예

◈ 텍스트 파일 내용 복사 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_copy file_copy.c target.c
[cprog2@seps5 ch10]$ ls -al
합계 36
drwxrwxr-x    2 cprog2   cprog2      4096  2월  10 23:03 .
drwxrwxr-x   12 cprog2   cprog2      4096  2월   3 11:32 ..
-rw-rw-r--    1 cprog2   cprog2       167  2월   9 16:47 README_by_LJM
-rwxrwxr-x    1 cprog2   cprog2     10561  2월  10 22:21 file_copy
-rw-rw-r--    1 cprog2   cprog2       719  2월  10 22:21 file_copy.c
-rw-rw-r--    1 cprog2   cprog2       719  2월  10 23:03 target.c
[cprog2@seps5 ch10]$
```

저수준 파일 처리 함수

- UNIX 파일 접근 프리미티브
  - 프로그램 안에서 파일들을 다루기 위해 UNIX 가 제공하는 기본적인 프리미티브들
  - UNIX 커널에 의해 제공되는 I/O 장치에 직접 접근을 제공하는 시스템 호출들의 작은 집합
  - UNIX 파일 접근 프리미티브들

이름	의 미
open	읽거나 쓰기 위해 파일을 열거나, 또는 빈 파일을 생성
creat	빈 파일을 생성한다
close	열려진 파일을 닫는다
read	파이로부터 정보를 추출한다
write	파일에 정보를 기록한다
lseek	파일 안의 지정된 바이트로 이동한다.
unlink	파일을 제거한다.
remove	파일을 제거하는 다른 방법
fcntl	한 파일에 연관된 속성을 제어한다.

저수준 파일 처리 함수

- 프로그램 예

```
/* 초보적인 프로그램 예 */

/* 이 헤더 파일들은 아래에서 논의한다 */
#include <fcntl.h>
#include <unistd.h>

main()
{
        int fd;
        ssize_t nread;
        char buf[1024];

        /* 파일 "data"를 읽기 위해 개방한다 */
        fd = open("data", O_RDONLY);

        /* 데이터를 읽어들인다 */
        nread = read(fd, buf, 1024);

        /* 파일을 폐쇄한다 */
        close(fd);
}
```

파일을 읽기 전용으로 개방

file descriptor

open 시스템 호출

- 기능
  - ◈ 기존의 파일을 읽거나 쓰기 전에 항상 파일 개방
- 사용법

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

int open(const char *pathname, int flags, [mode_t mode]);
```

- **pathname :** 개방될 파일의 경로
- **Flags :** 접근 방식 지정
  - **O_RDONLY** 읽기 전용으로 개방
  - **O_WRONLY** 쓰기 전용으로 개방
  - **O_RDWR** 읽기 및 쓰기 용으로 개방
  - 아래 상수는 위의 상수와 **OR** 해서 사용
  - **O_CREAT** 파일이 없으면 생성, 아래 **mode** 필요함
  - **O_APPEND** 파일 쓰기 시 파일 끝에 추가
  - **O_TRUNC** 파일이 이미 존재하고 쓰기 권한으로 열리면 크기를 **0**
- **Mode :** 보안과 연관, 생략 가능

open 시스템 호출

사용 예

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

#define PERMS 0644    /* O_CREAT를 사용하는 open 을 위한 허가 */
char *workfile="junk";

main()
{
        int filedes;

        if ((filedes = open(workfile, O_RDWR | O_CREAT, PERMS)) == -1)
        {
                printf ("Couldn't open %s₩n", workfile);
                exit (1);                    /* 오류이므로 퇴장한다 */
        }

        /* 프로그램의 나머지 부분이 뒤따른다 */

        exit (0);
}
```

close 시스템 호출

기능

- Open 의 역, 개방 중인 파일을 닫음

사용법

```
#include <unistd.h>

int close(int filedes);
```

- **filedes :** 닫혀질 파일 기술자

```
filedes = open("file", O_RDONLY);
.
.

close(filedes);
```

저수준 파일 처리 함수

▣ 파일 열기/닫기 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_check.c
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

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

    if (argc < 2) {
        fprintf(stderr, "Usage: file_check filename₩n");
        exit(1);
    }
    if ( (fd = open(argv[1], O_RDONLY)) == -1 ) {
        perror("open");  /* errno에 대응하는 메시지 출력됨 */
        exit(1);
    }

    printf("File ₩"%s₩" found...₩n", argv[1]);

    close(fd);
}
```

저수준 파일 처리 함수

▣ 파일 열기/닫기 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_check file_check.c
File "file_check.c" found...          <-- 27 줄에 해당하는 메시지 출력
[cprog2@seps5 ch10]$ file_check file_check.c2
open: No such file or directory       <-- 23 줄에 해당하는 메시지 출력
```

Open 시스템 호출

▣ 파일 생성(O_CREAT)시 접근 권한(mode) 지정

사용자 권한		그룹 권한		기타 사용자 권한	
S_IRWXU	읽기, 쓰기, 실행 가능	S_IRWXG	읽기, 쓰기, 실행 가능	S_IRWXO	읽기, 쓰기, 실행 가능
S_IRUSR	읽기 가능	S_IRGRP	읽기 가능	S_IROTH	읽기 가능
S_IWUSR	쓰기 가능	S_IWGRP	쓰기 가능	S_IWOTH	쓰기 가능
S_IXUSR	실행 가능	S_IXGRP	실행 가능	S_IXOTH	실행 가능

```
[cprog2@seps5 ch10]$ ls -al
합계 68
drwxrwxr-x   2 cprog2 cprog2     4096  2월 11 08:56 .
drwxrwxr-x 12 cprog2 cprog2     4096  2월  3 11:32 ..
-rwxrwxr-x   1 cprog2 cprog2    10502   2월 11 08:56 file_creat
-rw-rw-r--   1 cprog2 cprog2      727  2월 11 08:56 file_creat.c
-rw-r--r--   1 cprog2 cprog2       1 6   2월 11 08:56 t.txt
```

creat 시스템 호출

▣ 기능

◈ 파일을 생성하는 대안적 방법

▣ 사용법

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

int creat(const char *pathname, mode_t mode);
```

- **pathname :** 개방될 파일의 경로
- **Mode :** 필요한 접근 허가 제시, 생략 가능

```
filedes = creat("/tmp/newfile", 0644);
filedes = open("/tmp/newfile", O_WRONLY | O_CREAT | O_TRUNC, 0644);
```

저수준 파일 처리 함수

▣ 파일 생성 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_creat.c
 */

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

main(int argc, char *argv[])
{
    int fd;  /* file descriptor */
    char *buf = "This is a test.";
    ssize_t cnt;  /* write count */
    int flags = O_WRONLY | O_CREAT | O_TRUNC;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP
| S_IROTH; /* == 0644 */
    if (argc < 2) {
        fprintf(stderr, "Usage: file_creat filename₩n");
        exit(1);
    }

    if ( (fd = open(argv[1], flags, mode)) == -1 ){
        perror("open");  /* errno에 대응하는 메시지 출
력됨 */
        exit(1);
    }

    cnt = write(fd, buf, strlen(buf));
    printf("write count = %d₩n", cnt);
    close(fd);
}
```

저수준 파일 처리 함수

▣ 파일 생성 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_creat t.txt
write count = 16
[cprog2@seps5 ch10]$ ls -al
합계 68
drwxrwxr-x    2 cprog2   cprog2      4096  2월 11 08:56 .
drwxrwxr-x   12 cprog2   cprog2      4096  2월  3 11:32 ..
-rw-rw-r--    1 cprog2   cprog2       167  2월  9 16:47 README_by_LJM
-rwxrwxr-x    1 cprog2   cprog2     10259  2월 11 01:53 file_check
-rw-rw-r--    1 cprog2   cprog2       532  2월 11 01:53 file_check.c
-rwxrwxr-x    1 cprog2   cprog2     10561  2월 10 22:21 file_copy
-rw-rw-r--    1 cprog2   cprog2       719  2월 10 22:21 file_copy.c
-rwxrwxr-x    1 cprog2   cprog2     10502  2월 11 08:56 file_creat
-rw-rw-r--    1 cprog2   cprog2       727  2월 11 08:56 file_creat.c
-rw-r--r--    1 cprog2   cprog2        16  2월 11 08:56 t.txt
-rw-rw-r--    1 cprog2   cprog2       719  2월 10 23:03 target.c
[cprog2@seps5 ch10]$ cat t.txt
This is a test.
[cprog2@seps5 ch10]$
```

read 시스템 호출

기능

- 파일로부터 임의의 문자들 또는 바이트들을 호출 프로그램의 제어 하에 있는 버퍼로 복사

사용법

```
#include <unistd.h>

ssize_t read(int filedes, void *buffer, size_t n);
```

- **filedes :** 이전의 **open** 또는 **creat** 로부터 얻은 파일 기술자
- **buffer :** 자료가 복사되어질 문자 배열의 포인터
- **n :** 파일로부터 읽혀질 바이트의 수

읽기 쓰기 포인터

- 순차적인 파일 읽기 예제

```
int fd;
ssize_t n1, n2;
char buf1[512], buf2[512];
.
.
if ( (fd = open("foo", O_RDONLY) ) == -1)
   return (-1);

n1 = read (fd, buf1, 512);      /* 처음 512 문자들을 buf1 에 저장 */
n2 = read (fd, buf2, 512);      /* 다음 512 문자들을 buf2 에 저장 */
```

- 읽기 쓰기 포인터 또는 파일 포인터
  - **Read** 시스템 호출**:** 각 호출 후에 읽혀진 바이트 수만큼 읽기-쓰기 포인터를 전진
- **read** 로부터 **0** 이 반환되면 파일의 끝

한 파일 내의 문자 수를 세는 프로그램

■ 사용 예

```
/* count – 한 파일 내의 문자 수를 센다 */
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define     BUFSIZE    512          /* 읽혀질 덩어리의 크기 */

main()
{
   char buffer[BUFSIZE];
   int filedes;
   ssize_t nread;
   long total = 0;

   /* “anotherfile” 을 읽기 전용으로 개방 */
   if ( (filedes = open (“anotherfile”, O_RDONLY) ) == -1)
   {
      printf (“error in opening anotherfile\n”);
      exit (1);
   }

   /* EOF 까지 반복하라. EOF 는 복귀값 0 에 의해 표시된다 */
   while ( (nread = read (filedes, buffer, BUFSIZE) ) > 0)
      total += nread;                  /* total 을 증가시킨다 */

   printf (“total chars in anotherfile: %ld\n”, total);
   exit (0);
}
```

write 시스템 호출

▣ 기능

◈ 문자 배열인 프로그램 버퍼로부터 외부 파일로 자료를 복사

▣ 사용법

```
#include <unistd.h>

ssize_t write(int filedes, const void *buffer, size_t n);
```

- **filedes :** 이전의 **open** 또는 **creat** 로부터 얻은 파일 기술자
- **buffer :** 자료가 복사되어질 문자 배열의 포인터
- **n :** 파일로 쓰여질 바이트의 수

읽기 쓰기 포인터

순차적인 파일 쓰기 예제

```
int fd;
ssize_t w1, w2;
char header1[512], header2[1024];
.
.
if ( (fd = open("newfile", O_WRONLY | O_CREAT | O_EXCL, 0664) ) == -1)
    return (-1);

w1 = write (fd, header1, 512);          /* header1의 512문자들을 파일에 씀 */
w2 = write (fd, header2, 1024);         /* header2의 1024문자들을 파일에 연속해서 씀*/
```

읽기 쓰기 포인터

- **write** 시스템 호출: 각 호출 후에 쓰여진 바이트 수만큼 읽기-쓰기 포인터를 전진
- **O_APPEND** 를 사용하면 파일의 마지막 바이트 뒤에 위치

저수준 파일 처리 함수

▣ 파일 추가 모드 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_append.c
 */

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

main(int argc, char *argv[])
{
    int fd;  /* file descriptor */
    char *buf = "This is a test.₩n";
    ssize_t cnt;  /* write count */
    int flags = O_WRONLY | O_CREAT | O_APPEND;
    mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP
| S_IROTH; /* == 0644 */

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

    if ( (fd = open(argv[1], flags, mode)) == -1 ){
        perror("open");  /* errno에 대응하는 메시지
출력됨 */
        exit(1);
    }

    cnt = write(fd, buf, strlen(buf));
    printf("write count = %d₩n", cnt);

    close(fd);
}
```

저수준 파일 처리 함수

파일 추가 모드 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ cat > t.txt      <-- 테스트 파일 “t.txt” 생성
File "t.txt" exists...
^D
```

```
[cprog2@seps5 ch10]$ file_append t.txt
write count = 16
[cprog2@seps5 ch10]$ cat t.txt
File "t.txt" exists...
This is a test.              <-- file_append.c 31번줄에 의하여 추가된 내용
[cprog2@seps5 ch10]$
```

저수준 파일 처리 함수

▣ 파일 복사 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_copy.c
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#define MAX_READ 10

main(int argc, char *argv[])
{
    int src_fd;  /* source file descriptor */
    int dst_fd;  /* destination file descriptor */
    char buf[MAX_READ];
    ssize_t rcnt;       /* read count */
    ssize_t tot_cnt = 0;  /* total write count */
    mode_t mode = S_IRUSR | S_IWUSR |
S_IRGRP | S_IROTH; /* == 0644 */

    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"); /* errno에 대응하는 메시지 출
력됨 */
        exit(1);
    }
    if ( (dst_fd = creat(argv[2], mode)) == -1 ){
        perror("dst open");  /* errno에 대응하는 메시지 출
력됨 */
        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 = %d₩n", tot_cnt);
   close(src_fd);
   close(dst_fd);
}
```

저수준 파일 처리 함수

▣ 파일 복사 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_copy file_copy.c target.c
total write count = 1137
[cprog2@seps5 ch10]$ diff file_copy.c target.c
[cprog2@seps5 ch10]$          <-- diff 실행 결과 두 파일이 동일함을 알 수 있다.
```

▣ 파일 접근 프리미티브의 효율성 측정

- ◈ **Filecopy.c 프로그램 이용**
- ◈ 측정 결과 - 버퍼 크기가 클수록 효율적, 즉 시스템 호출의 비용

BUFSIZE	실제시간	사용자시간	시스템시간
1	0:24.49	0: 3.13	0:21.16
64	0: 0.46	0: 0.12	0: 0.33
512	0: 0.12	0: 0.02	0: 0.08
4096	0: 0.07	0: 0.00	0: 0.05
8192	0: 0.07	0: 0.01	0: 0.05

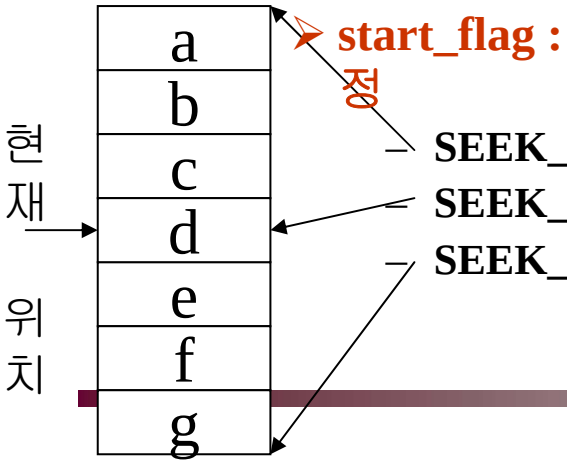
lseek 와 임의 접근

- 기능
  - 읽기쓰기 포인터의 위치 (다음에 읽거나 쓸 바이트 위치) 변경
  - 파일에 대한 임의 접근 가능
- 사용법

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

off_t lseek(int filedes, off_t offset, int start_flag);
```

- **filedes :** 개방되어 있는 파일의 파일 기술자
- **offset :** 읽기-쓰기 포인터의 새 위치 결정
- **start_flag :** 읽기-쓰기 포인터가 파일의 어느 지점을 시작으로 할 지 결정
  - **SEEK_SET** **offset** 을 파일의 시작부터 계산
  - **SEEK_CUR** **offset** 을 파일 포인터의 현재 위치부터 계산
  - **SEEK_END** **offset** 을 파일의 끝부터 계산



lseek 시스템 호출

- **offset** 은 음수 가능
  - 시작점으로부터 거꾸로 이동 가능
- 기존 파일의 끝에 추가하여 쓰는 예제

```
filedes = open (filename, O_RDWR);
lseek (filedes, (off_t) 0, SEEK_END);
write (filedes, outbuf, OBSIZE);
```

  - 또 다른 방법 : **O_APPEND** 추가하여 **open** 호출
- 파일의 크기 알아내는 예제

```
off_t filesize;
int filedes;
.
.
filesize = lseek (filedes, (off_t) 0, SEEK_END);
```

저수준 파일 처리 함수

▣ 파일 크기 알아내기 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_size.c
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

main(int argc, char *argv[])
{
    int fd;  /* file descriptor */
    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 = %d₩n", argv[1], size);

    close(fd);
}
```

저수준 파일 처리 함수

- 파일 크기 알아내기 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_size file_size.c
file_size.c's size = 643
[cprog2@seps5 ch10]$ ls -al file_size.c
-rw-rw-r--    1 cprog2   cprog2        643  2월 11 16:34 file_size.c
```

저수준 파일 처리 함수

▣ 파일 크기 변경 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_hole.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

const char *endstring = ".";

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

    if (argc < 3) {
        fprintf(stderr, "Usage: file_hole filename size₩n");
        exit(1);
    }

    if ( (fd = open(argv[1], O_RDWR)) == -1 ){
        perror("open");  /* errno에 대응하는 메시지 출력됨 */
        exit(1);
    }

    hole_size = atoi(argv[2]);
    size = lseek(fd, 0, SEEK_END);
    printf("Before : %s₩'s size = %d₩n", argv[1], size);
    lseek(fd, hole_size, SEEK_END);
    write(fd, endstring, 1); /* 오프셋이 파일 크기를 초과하는 경우    */
        /* write()가 있어야만 NULL 문자로 채운다. */
    size = lseek(fd, 0, SEEK_END);
    printf("After : %s₩'s size = %d₩n", argv[1], size);
    close(fd);
}
```

저수준 파일 처리 함수

▣ 파일 크기 변경 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ ls -al test.txt
-rw-r--r--    1 cprog2   cprog2        16  2월 11 18:59 test.txt
[cprog2@seps5 ch10]$ od -c test.txt
0000000   T   h   i   s       i   s       a       t   e   s   t   .  ₩n
0000020
[cprog2@seps5 ch10]$ file_hole test.txt 10
Before : test.txt's size = 16
After  : test.txt's size = 27
[cprog2@seps5 ch10]$ ls -al test.txt
-rw-r--r--    1 cprog2   cprog2        27  2월 11 19:00 test.txt
[cprog2@seps5 ch10]$ od -c test.txt
0000000   T   h   i   s       i   s       a       t   e   s   t   .  ₩n
0000020  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0  ₩0   .
0000033
```

표준 입력, 표준 출력 및 표준 오류

- **UNIX** 시스템은 수행중인 한 프로그램에 대해 자동적으로 세 개의 파일을 개방
  - 표준 입력 **(0),** 표준 출력 **(1),** 표준 오류**(2)**
  - **# prog_1 | prog_2 : prog_1** 의 표준 출력을 **prog_2** 의 표준 입력으로 사용
  - 표준 입력과 출력 파일 기술자는 유연하고 일관된 프로그램 구성 수단
- 표준 입력
  - 디폴트로 키보드로부터 자료를 입력받음
  - **# prog_name < infile**
- 표준 출력
  - 디폴트로 단말기 화면으로 자료를 출력
  - **# prog_name > outfile**

표준 입력, 표준 출력 및 표준 오류

표준 입출력 예제 프로그램

```
/* 8장 파일 처리
 * 파일 이름: file_io.c
 */

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

#define BUFSIZE 256

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);
}
```

```
[cprog2@seps5 cprog2]$ file_io
Here is file start.
This is a test message.          <--- 키보드로부터 입력
This is a test message.^D
Here is file end.
[cprog2@seps5 cprog2]$ file_io < test.txt > test1.txt
Here is file start.
Here is file end.
[cprog2@seps5 cprog2]$ cat test1.txt
This is a test.
[cprog2@seps5 cprog2]$
```

표준 입력, 표준 출력 및 표준 오류

▣ 표준 오류

- ◈ 오류와 경고 메시지를 위한 특별한 출력 채널
- ◈ **# make > log.out 2> log.err**
  - ➢ **Log.out :** 표준 출력 결과**, log.err :** 표준 오류 결과
- ◈ 시스템 호출 **write** 와 파일 기술자 **2** 를 사용

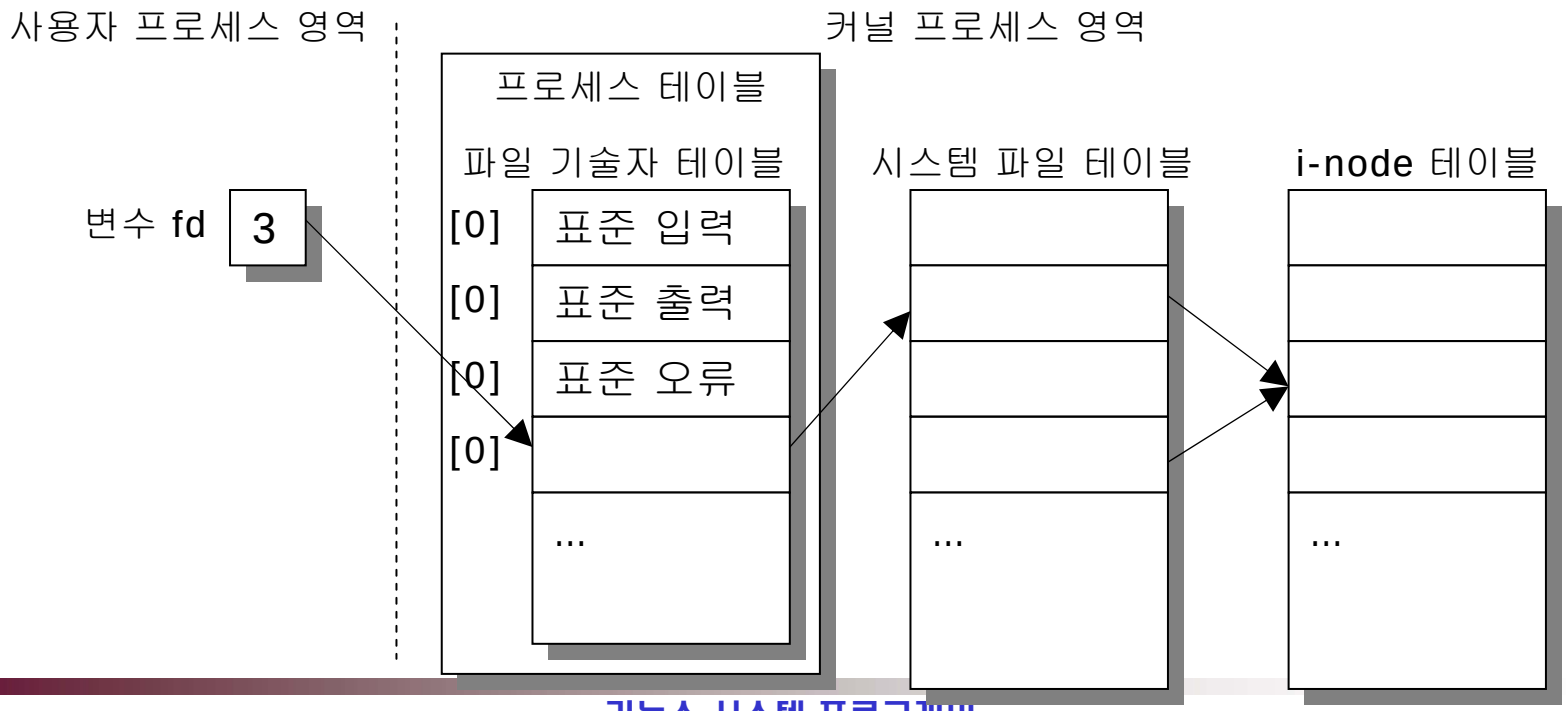
```
char msg[6] = “boob\n”;
.
.
write (2, msg, 5);
```

파일의 표현

▣ 시스템의 파일 관리

◈ 파일 기술자 테이블, 시스템 파일 테이블, **i-node** 테이블을 통해 각 프로세스가 사용하는 파일 관리

- ➢ 시스템 파일 테이블 – 시스템의 모든 파일에 대한 정보(상태, 현재 오프셋 등) 수록
- ➢ **i-node** 테이블 – 실제 저장된 파일의 정보 수록



파일 기술자 복사 함수

기능

- 파일 기술자를 새로운 항목으로 복사
  - **dup** 함수는 소스 파일 기술자를 사용하지 않는 가장 작은 파일 기술자로 복사
  - **dup2** 함수는 소스 파일 기술자를 대상 파일 기술자로 복사
- 복사되기 전후의 기술자들은 같은 시스템 파일 항목 공유
  - 락, 파일 위치 포인터, 플래그 공유

사용법

```
#include <unistd.h>

int dup (int filedes);
int dup2 (int filedes, int filedes2);
```

- **filedes :** 존재하는 소스 파일 기술자 항목
- **filedes2:** 복사할 대상 파일 기술자 항목
- 반환되는 값은 새로운 파일 기술자 번호, 복사가 실패하면 **-1** 반환

파일 기술자 복사 함수

▣ 표준 출력 재지향 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_output.c
 */
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

main(int argc, char *argv[])
{
    int fd;  /* file descriptor */
    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; /* == 0644 */

    if (argc < 2) {
        fprintf(stderr, "Usage: file_output
filename₩n");
        exit(1);
    }
    if ( (fd = open(argv[1], flags, mode)) == -1 ) {
        perror("open");  /* errno에 대응하는 메시지
출력됨 */
        exit(1);
    }
    if ( dup2(fd, 1) == -1 ) {
       perror("dup2");  /* errno에 대응하는 메시지
출력됨 */
        exit(1);
    }
    if ( close(fd) == -1 ) {
        perror("close");  /* errno에 대응하는 메시지
출력됨 */
        exit(1);
    }
    write(1, buf, strlen(buf));
    exit(0);
}
```

파일 기술자 복사 함수

- 표준 출력 재지향 프로그램 실행 결과

```
[cprog2@seps5 ch8]$ file_output
Usage: file_output filename
[cprog2@seps5 ch8]$ file_output output.txt
[cprog2@seps5 ch8]$ cat output.txt
This is a test output file.
[cprog2@seps5 ch8]$
```

파일 상태

- 기능
  - 파일 유형 등의 상태 정보 처리
- 사용법

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

int stat(const char *filename, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat(const char *filename, struct stat *buf);
```

- **stat() : filename** 이 주어지면 해당 파일에 대한 정보 획득
- **fstat() :** 파일 기술자를 인자로 파일 정보 획득
- **lstat() : stat()** 와 비슷하나 해당 파일이 심볼릭 링크인 경우 링크의 정보 획득
- **buf :** 파일 정보를 정의한 **stat** 구조체 포인터

파일 상태

- **LINUX stat 구조체**
  - 파일 정보 저장
  - **<bits/stat.h>** 에 정의

```
struct stat {
        dev_t         st_dev;       /* device */
        ino_t         st_ino;       /* inode */
        mode_t        st_mode;      /* protection */
        nlink_t       st_nlink;     /* number of hard links */
        uid_t         st_uid;       /* user ID of owner */
        gid_t         st_gid;       /* group ID of owner */
        dev_t         st_rdev;      /* device type (if inode device) */
        off_t         st_size;      /* total size, in bytes */
        unsigned long  st_blksize;  /* blocksize for filesystem I/O */
        unsigned long  st_blocks;   /* number of blocks allocated */
        time_t        st_atime;     /* time of last access */
        time_t        st_mtime;     /* time of last modification */
        time_t        st_ctime;     /* time of last change */
};
```

- 파일 유형을 테스트하는 매크로들
  - **<sys/stat.h>** 에 정의

매크로	의 미
S_ISREG()	Regular file (일반적인 파일 형태)
S_ISDIR()	Directory file (다른 파일을 가지는 파일 형태)
S_ISCHR()	Character special file (문자 장치에 사용)
S_ISBLK()	Block special file (블록 장치에 사용)
S_ISFIFO()	FIFO pipe (프로세스간 IPC 에 사용)
S_ISLNK()	Symbolic link file (다른 파일에 대한 링크 파일 형태)
S_ISSOCK()	socket file (네트워크 통신에 사용)

저수준 파일 처리 함수

▣ 파일 형태 출력 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_stat.c
 */

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

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

    if (argc < 2) {
        fprintf(stderr, "Usage: file_stat filename₩n");
        exit(1);
    }
```

```
    if (lstat(argv[1], &buf) < 0) {
        perror("stat");
        exit(1);
    }

    // Check the file type
    if (S_ISREG(buf.st_mode)) msg = "regular file";
    else if (S_ISDIR(buf.st_mode)) msg = "directory";
    else if (S_ISCHR(buf.st_mode)) msg = "character
special file";
    else if (S_ISBLK(buf.st_mode)) msg = "block
special file";
    else if (S_ISFIFO(buf.st_mode)) msg = "pipe of
FIFO";
    else if (S_ISLNK(buf.st_mode)) msg = "symbolic
link";
    else if (S_ISSOCK(buf.st_mode)) msg = "socket";

    printf("₩"%s₩" is %s₩n", argv[1], msg);
}
```

저수준 파일 처리 함수

▣ 파일 형태 출력 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ ln -s test.txt t2.txt
[cprog2@seps5 ch10]$ ls -al *.txt
lrwxrwxrwx    1 cprog2   cprog2          8  2월 12 00:28 t2.txt -> test.txt
-rw-r--r--    1 cprog2   cprog2         27  2월 11 19:00 test.txt
[cprog2@seps5 ch10]$ file_stat test.txt
"test.txt" is regular file
[cprog2@seps5 ch10]$ file_stat t2.txt
"t2.txt" is symbolic link
[cprog2@seps5 ch10]$ file_stat ~/book2/ch10
"/home/cprog2/book2/ch10" is directory
```

access 시스템 호출

기능

- 실제 사용자 ID 와 실제 그룹 ID 에 기반해서 파일에 대한 접근성 검사

사용법

```
#include <unistd.h>

int access(const char *pathname, int mode);
```

- **pathname :** 접근성 검사할 파일에 대한 경로
- **mode :** 검사할 파일의 모드, 다음 매크로들을 **bitwise OR**
  - R_OK : 읽기 권한에 대한 검사
  - W_OK : 쓰기 권한에 대한 검사
  - X_OK : 실행 권한에 대한 검사
  - F_OK : 파일 존재에 대한 검사

저수준 파일 처리 함수

▣ 파일 접근성 검사 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름 : file_access.c
 */

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

main(int argc, char *argv[])
{
    int ret_val;
    int mode_array[4] = {F_OK, R_OK, W_OK, X_OK};
    const char *mode_str[4] = {"existent", "readble",
                "writable", "executable"};
    const char *mode_err[4] = {"Existence
check", "Readability check",
                "Writability check", "Executability
check"};
    int i;

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

    printf("File ₩"%s₩" : ", argv[1]);

    for (i=0; i<4; i++) {
        ret_val = access(argv[1], mode_array[i]);
        if (ret_val == 0)
            printf("%s ", mode_str[i]);
        else if (ret_val == -1) {
            fprintf(stderr, "%s: %s₩n", mode_err[i],
strerror(errno));
        }
    }
    printf("₩n");
}
```

저수준 파일 처리 함수

▣ 파일 접근성 검사 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_access file_access
File "file_access" : existent readble writable executable
[cprog2@seps5 ch10]$ file_access file_access.c
Executability check: Permission denied          <-- stderr로 출력된 메시지
File "file_access.c" : existent readble writable
[cprog2@seps5 ch10]$ file_access /usr/include
Writability check: Permission denied            <-- stderr로 출력된 메시지
File "/usr/include" : existent readble executable
[cprog2@seps5 ch10]$ file_access /usr/include777
Existence check: No such file or directory      <-- stderr로 출력된 메시지
Readability check: No such file or directory    <-- stderr로 출력된 메시지
Writability check: No such file or directory    <-- stderr로 출력된 메시지
Executability check: No such file or directory <-- stderr로 출력된 메시지
File "/usr/include777" :
[cprog2@seps5 ch10]$
```

umask 시스템 호출

▣ 기능

◈ 파일 모드 생성 매스크를 설정하고 이전 값을 반환

▣ 사용법

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

mode_t umask(mode_t mask);
```

- **mask :** 파일 모드 생성 매스크
- 예) **umask** 기본값이 **022** 인 경우 **mode** 가 **0666** 이면 **0666& ~022 = 0644 = rw-r--r–** 허가권으로 파일 생성

저수준 파일 처리 함수

▣ 파일 권한 제어 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_umask.c
 */

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

main(int argc, char *argv[])
{
    int mode = S_IRUSR | S_IWUSR | S_IRGRP |
S_IWGRP | S_IROTH | S_IWOTH;

    umask(0);
    if (creat("test1.txt", mode) < 0) {
        perror("creat: test1.txt");
        exit(1);
    }
    umask(S_IRGRP | S_IWGRP | S_IROTH |
S_IWOTH);
    if (creat("test2.txt", mode) < 0) {
        perror("creat: test1.txt");
        exit(1);
    }
}
```

저수준 파일 처리 함수

파일 권한 제어 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_umask
[cprog2@seps5 ch10]$ ls -al test?.txt
-rw-rw-rw-    1 cprog2   cprog2          0  2월 14 14:42 test1.txt
-rw-------    1 cprog2   cprog2          0  2월 14 14:42 test2.txt
```

디렉토리 관련 함수

- 기능
  - 디렉토리 관련 동작 수행
- 사용법

```
#include <sys/types.h>
#include <dirent.h>

DIR *opendir(const char *name);
int closedir(DIR *dir);
struct dirent *readdir(DIR *dir);
void rewinddir(DIR *dir);
```

- **opendir() :** 디렉토리 **name** 에 해당하는 디렉토리 스트림을 개방하고, 디렉토리 스트림에 대한 포인터 반환
- **closedir() : dir** 과 연관된 디렉토리 스트림을 닫는다
- **readdir() :** 디렉토리 항목을 읽고, 이를 나타내는 **dirent** 구조체 반환
- **rewinddir() :** 디렉토리 스트림 **dir** 의 위치를 디렉토리의 처음으로 리셋

디렉토리 관련 함수

LINUX dirent 구조체

- 디렉토리 정보 저장 – 구현 시스템마다 다름
- **<bits/dirent.h>** 에 정의

```
struct dirent {
#ifndef __USE_FILE_OFFSET64
    __ino_t d_ino;
    __off_t d_off;
#else
    __ino64_t d_ino;
    __off64_t d_off;
#endif
    unsigned short int d_reclen;
    unsigned char d_type;
    char d_name[256];
  };
```

디렉토리 관련 함수

▣ 디렉토리 내용 보기 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_dir.c
 */

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

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);
}
```

디렉토리 관련 함수

디렉토리 내용 보기 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ ls -a /usr
.  X11R6 dict etc   hwpx   kerberos libexec man   share tmp
.. bin   doc  games include lib      local   sbin  src
[cprog2@seps5 ch10]$ file_dir /usr
                .                ..             share
            X11R6               bin              dict
              etc             games           include
              lib             local              sbin
              src           libexec               tmp
          kerberos               doc               man
             hwpx
```

getcwd 함수

- 기능
  - 작업 디렉토리에 대한 정보 읽기
- 사용법

```
#include <unistd.h>

char *getcwd(char *buf, size_t size);
```

  - **buf :** 현재 작업 디렉토리의 절대 경로를 복사할 버퍼 포인터
  - **size :** 경로 이름이 초과할 경우 **NULL** 이 반환

chdir 함수

- 기능
  - 작업 디렉토리 변경
- 사용법

```
#include <unistd.h>

int chdir(const char *path);
```

  - **path :** 현재 디렉토리를 변경할 경로

mkdir 함수

- 기능
  - 디렉토리 생성
- 사용법

```
#include <unistd.h>

int mkdir(const char *pathname, mode_t mode);
```

  - **pathname :** 생성하려고 시도하는 디렉토리 경로
  - **mode :** 생성 시 사용할 수 있는 권한에 대한 허가권 지정

rmdir 함수

기능

- 디렉토리 삭제, 이 때 디렉토리는 비어 있어야 한다.

사용법

```
#include <unistd.h>

int rmdir(const char *pathname);
```

- **pathname :** 삭제할 디렉토리 경로

디렉토리 관련 함수

▣ 디렉토리 이동 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_chdir.c
 */

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

#define MAX_BUF 256

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);
}
```

디렉토리 관련 함수

디렉토리 이동 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ file_chdir /usr
working directory (before) = /home/cprog2/book2/ch10
working directory (after ) = /usr
[cprog2@seps5 ch10]$ file_chdir ..
working directory (before) = /home/cprog2/book2/ch10
working directory (after ) = /home/cprog2/book2
```

```
[cprog2@seps5 ch10]$ file_chdir /root
working directory (before) = /home/cprog2/book2/ch10
chdir: Permission denied            <-- 접근 권한이 없음.
```

기타 함수

▣ 파일 링크 관련 함수

◈ 기능

- 파일에 대한 링크와 관련된 동작 수행

◈ 사용법

```
#include <unistd.h>

int link(const char *oldpath, const char *newpath);
int unlink(const char *pathname);
int symlink(const char *oldpath, const char *newpath);
```

- **link() : oldpath** 파일에 대한 **newpath** 하드 링크를 생성
- **unlink() :** 파일 시스템에서 **pathname** 파일 삭제. 파일이 심볼릭 링크이면 링크를 삭제
- **symlink() : oldpath** 파일에 대한 **newpath** 심볼릭 링크를 생성

기타 함수

▣ 파일 링크 생성/삭제 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_link.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main(int argc, char *argv[])
{
    char cmd;
    char *src;
    char *target;

    if (argc < 3) {
      fprintf(stderr, "Usage: file_link [l,u,s] ...₩n");
      fprintf(stderr, "       file_link l[ink] src
target₩n");
      fprintf(stderr, "    file_link u[nlink] filename₩n");
      fprintf(stderr, "   file_link s[ymlink] src
target₩n");
        exit(1);
    }
    cmd = (char) *argv[1];
    if ( cmd == 'l' ) {
        if (argc < 4) {
            fprintf(stderr, "file_link l src target [link]₩n");
            exit(1);
        }
        src = argv[2];
        target = argv[3];
        if (link(src, target) < 0) {
            perror("link");
            exit(1);
        }
    }
    else if (cmd == 's') {
        if (argc < 4) {
            fprintf(stderr, "file_link l src target [link]₩n");
            exit(1);
        }
        src = argv[2];
        target = argv[3];
```

기타 함수

- 파일 링크 생성/삭제 프로그램 계속

```
        if (symlink(src, target) < 0) {
            perror("symlink");
            exit(1);
        }
    }
    else if (cmd == 'u') {
        src = argv[2];
        if (unlink(src) < 0) {
            perror("unlink");
            exit(1);
        }
    }
    else {
        fprintf(stderr, "Unknown command...₩n");
    }
}
```

기타 함수

▣ 파일 링크 생성/삭제 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ cat > test.txt
This is a test…
```

```
[cprog2@seps5 ch10]$ file_link l test.txt t2.txt
[cprog2@seps5 ch10]$ ls -al test.txt t2.txt
-rw-rw-r--    2 cprog2   cprog2        18  2월 17 11:38 t2.txt
-rw-rw-r--    2 cprog2   cprog2        18  2월 17 11:38 test.txt
[cprog2@seps5 ch10]$ diff test.txt t2.txt
```

```
[cprog2@seps5 ch10]$ file_link u test.txt
[cprog2@seps5 ch10]$ ls -al test.txt t2.txt
ls: test.txt: 그런 파일이나 디렉토리가 없음
-rw-rw-r--    1 cprog2   cprog2        18  2월 17 11:38 t2.txt
[cprog2@seps5 ch10]$ cat t2.txt
This is a test…
```

기타 함수

▣ 파일 링크 생성/삭제 프로그램 실행 결과 계속

```
[cprog2@seps5 ch10]$ mv t2.txt test.txt
[cprog2@seps5 ch10]$ ls -al test.txt t2.txt
ls: t2.txt: 그런 파일이나 디렉토리가 없음
-rw-rw-r--    1 cprog2   cprog2         18  2월 17 11:38 test.txt
```

```
[cprog2@seps5 ch10]$ file_link s test.txt t2.txt
[cprog2@seps5 ch10]$ ls -al test.txt t2.txt
lrwxrwxrwx    1 cprog2   cprog2          8  2월 17 11:42 t2.txt -> test.txt
-rw-rw-r--    1 cprog2   cprog2         18  2월 17 11:38 test.txt
[cprog2@seps5 ch10]$ diff test.txt t2.txt
```

```
[cprog2@seps5 ch10]$ file_link u test.txt
[cprog2@seps5 ch10]$ ls -al test.txt t2.txt
ls: test.txt: 그런 파일이나 디렉토리가 없음
lrwxrwxrwx    1 cprog2   cprog2          8  2월 17 11:42 t2.txt -> test.txt
[cprog2@seps5 ch10]$ cat t2.txt
cat: t2.txt: 그런 파일이나 디렉토리가 없음
```

remove 함수

- 기능
  - 한 파일을 시스템으로부터 제거
- 사용법

```
#include <stdio.h>

int remove(const char *pathname);
```

- **pathname :** 제거될 파일의 경로, 이 때 **pathname** 이 파일을 가리키면 **unlink()** 함수가 호출되고 디렉토리이면 **rmdir()** 함수가 호출.

rename 함수

기능

- 파일의 위치/이름 변경

사용법

```
#include <stdio.h>

int rename(const char *oldpath, const char *newpath);
```

- **oldpath :** 바꾸거나 이동할 원래 파일 경로
- **newpath :** 바뀌거나 이동한 후 만들어지는 파일 경로

기타 함수

mv 명령어 구현 프로그램

```
/*
 * 8장 파일 처리
 * 파일 이름: file_rename.c
 */
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

main(int argc, char *argv[])
{
    struct stat buf;
    char *target;
    char *src_file_name_only;

    if (argc < 3) {
        fprintf(stderr, "Usage: file_rename src
target₩n");
        exit(1);
    }
```

```
    // Check argv[1] (src) whether it has directory
info or not.
    if (access(argv[1], F_OK) < 0) {
        fprintf(stderr, "%s not exists₩n", argv[1]);
        exit(1);
    }
    else {
        char *slash = strrchr(argv[1], '/');
        src_file_name_only = argv[1];
        if (slash != NULL) { // argv[1] has directory
info.
            src_file_name_only = slash + 1;
        }
    }

    // Make target into a file name if it is a directory
    target = (char *)calloc(strlen(argv[2])+1,
sizeof(char));
    strcpy(target, argv[2]);
```

기타 함수

mv 명령어 구현 프로그램 계속

```
    if (access(argv[2], F_OK) == 0) {
        if (lstat(argv[2], &buf) < 0) {
            perror("lstat");
            exit(1);
        }
        else {
            if (S_ISDIR(buf.st_mode)) {
                free(target);
                target = (char *)
calloc(strlen(argv[1])+strlen(argv[2])+2,
                                sizeof(char));
                strcpy(target, argv[2]);
                strcat(target, "/");
                strcat(target, src_file_name_only);
            }
        }
    }
    printf("source = %s₩n", argv[1]);
    printf("target = %s₩n", target);
    if (rename(argv[1], target) < 0) {
        perror("rename");
        exit(1);
    }
    free(target);
}
```

기타 함수

mv 명령어 구현 프로그램 실행 결과

```
[cprog2@seps5 ch10]$ cat > foo.txt
this is a test for rename()…
[cprog2@seps5 ch10]$ file_rename foo.txt bar.txt
source = foo.txt
target = bar.txt
[cprog2@seps5 ch10]$ ls -al foo.txt bar.txt
ls: foo.txt: 그런 파일이나 디렉토리가 없음
-rw-rw-r--    1 cprog2   cprog2         31  2월 15 00:25 bar.txt
```

```
[cprog2@seps5 ch10]$ mkdir test
[cprog2@seps5 ch10]$ file_rename bar.txt test
source = bar.txt
target = test/bar.txt
[cprog2@seps5 ch10]$ ls -al bar.txt test/bar.txt
ls: bar.txt: 그런 파일이나 디렉토리가 없음
-rw-rw-r--    1 cprog2   cprog2         31  2월 15 00:25 test/bar.txt
```

기타 함수

mv 명령어 구현 프로그램 실행 결과 계속

```
[cprog2@seps5 ch10]$ file_rename test/bar.txt ~
source = test/bar.txt
target = /home/cprog2/bar.txt
[cprog2@seps5 ch10]$ ls -al test/bar.txt ~/bar.txt
ls: test/bar.txt: 그런 파일이나 디렉토리가 없음
-rw-rw-r--    1 cprog2   cprog2        31  2월 15 00:25 /home/cprog2/bar.txt
```

```
[cprog2@seps5 ch10]$ file_rename test test2
source = test
target = test2
[cprog2@seps5 ch10]$ ls -al test test2
ls: test: 그런 파일이나 디렉토리가 없음
test2:
합계 8
drwxrwxr-x    2 cprog2   cprog2       4096  2월 15 00:32 .
drwxrwxr-x    3 cprog2   cprog2       4096  2월 15 00:32 ..
```

fcntl 함수

▣ 기능

◈ 개방된 파일에 대한 제어

▣ 사용법

```
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.h>

/* 주의: 마지막 인수의 타입은 생략부호 "..." 에 의해 표시된 대로 가변적 */
int fcntl(int filedes, int cmd, ...);
```

- **filedes :** 제어할 파일 기술자
- **cmd :** 제어에 대한 특정 기능
  - F_GETFL: open 에 의해 설정된 현재 파일의 상태 플래그 반환
  - F_SETFL: 파일에 연관된 상태 플래그를 재지정
- 세번째 인수부터는 인수 층 에 따라 좌우

fcntl 함수

fcntl 사용 예제 프로그램

```
/* filestatus –- 파일의 현재 상태를 기술한다 */
#include <fcntl.h>

int filestatus (int filedes)
{
    int arg1;

    if (( arg1 = fcntl (filedes, F_GETFL)) == -1)
    {
        printf (“filestatus failed₩n”);
        return (-1);
    }

    printf (“File descriptor %d: “, filedes);
    /* 개방 시의 플래그를 테스트한다 */
    switch (arg1 & O_ACCMODE) {
    case O_WRONLY:
        printf (“write-only”);
        break;
    case O_RDWR:
        printf (“read-write”);
        break;
    case O_RDONLY:
        printf (“read-only”);
        break;
    default:
        printf (“No such mode”);
    }

    if (arg1 & O_APPEND)
        printf (“ – append flag set”);
    printf (“₩n”);
    return (0);
}
```