프로그래밍 기초

15장. 파일 처리 Part 2 (이진 파일)

학습목표

- 텍스트 파일과 이진 파일의 차이를 이해하고 설명할 수 있다.
  - 주기억장치와 파일의 차이
  - 텍스트 파일과 이진 파일의 정의와 예
  - 파일 스트림과 파일모드의 이해
  - 함수 fopen()과 fclose()의 사용
- 텍스트 파일의 입출력 함수를 이해하고 설명할 수 있다.
  - 함수 fprintf()와 fscanf(), fscanf_s()의 사용
  - 함수 fputc()와 fgetc()의 사용
- 이진 파일의 입출력 함수를 이해하고 설명할 수 있다.
  - 함수 fwrite()와 fread()의 사용
- 파일 삭제 및 이름 바꾸기 함수를 이해하고 설명할 수 있다.
  - 함수 remove()와 rename()의 사용

문자열 처리 함수: sscanf(), sprintf()

- **sscanf(): 문자열에서 서식화된 입력을 받는 함수**

```
int sscanf(const char *str, const char *format, ...);
```

```
char *token = "25 54.32 Thompson 56789";
int n1, n2;
char name1[20];
double d1;

sscanf(token, "%d %lf %s %d", &n1, &d1, name1, &n2);
```

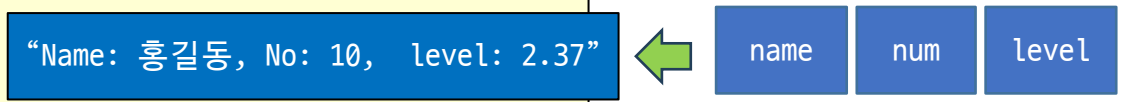
문자열에서 공백 기준으로 각 변수에 입력: sscanf(buf)

- **sprintf(): 버퍼에 포맷을 지정하여 출력(저장)하는 함수**

```
int sprintf(char *str, const char *format, ...);
```

```
char name[20] = "홍길동";
int num = 23;
double level = 2.37;

sprintf(buf, "Name: %s, No: %d, Level: %f", name, num, level);
```

각 변수들의 값을 하나의 문자열로 생성: sprintf(buf)



sscanf(), sprintf() 예제 (실습)

<sprintf_sscanf.c>

```c
#include <stdio.h>

int main()
{
    char *token = "25 54.32 Thompson 56789";
    int n1, n2;
    char name1[20];
    double d1;

    sscanf(token, "%d %lf %s %d", &n1, &d1, name1, &n2);
    printf("%d, %lf, %s, %d\n", n1, d1, name, n2);

    char buf[256];
    char name[20] = "홍길동";
    int num = 23;
    double level = 2.37;

    sprintf(buf, "Name: %s, No: %d, Level: %f", name, num, level);
    puts(buf);

    return 0;
}
```

실행 결과

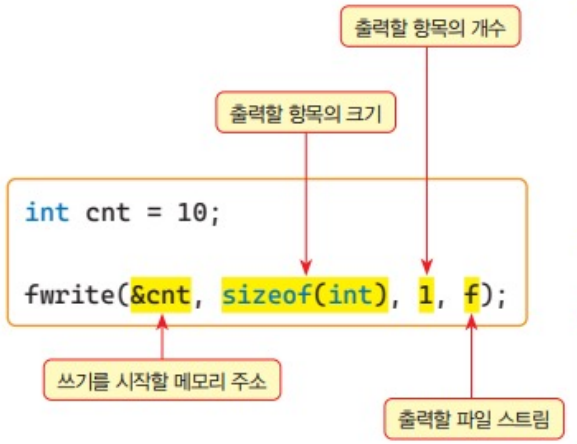
```
Name: 홍길동, No: 23, Level: 2.370000
25, 54.320000, 홍길동, 56789
```

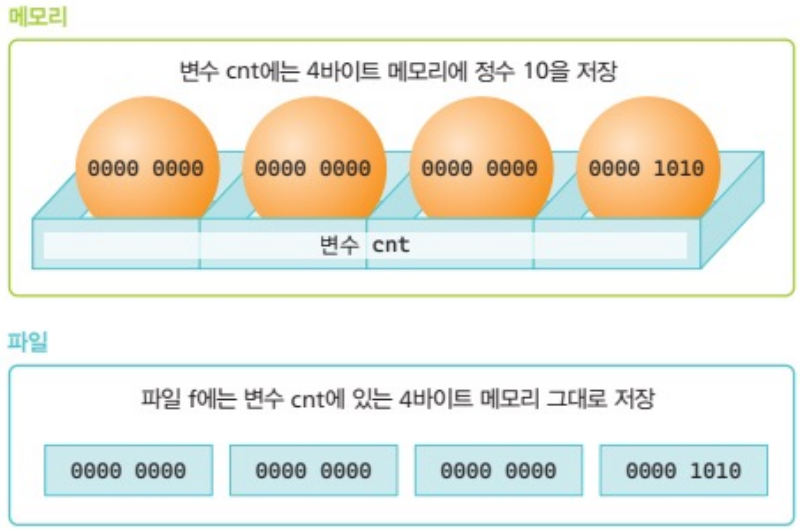
이진 파일 읽고 쓰기: fread(), fwrite()

- **함수 fwrite()와 fread()**
  - 이진 모드로 블록 단위 입출력을 처리
  - 이진 파일(binary file)
    - C 언어의 자료형을 모두 유지하면서 바이트 단위로 저장되는 파일
  - 헤더파일 stdio.h 필요

```
size_t fwrite(const void *ptr, size_t size, size_t nitems, FILE *stream);
size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
```

```
int cnt = 10;

fwrite(&cnt, sizeof(int), 1, f);
fread(&cnt, sizeof(int), 1, f);
```





함수 fwrite()와 fread()

- **함수 fwrite(): 바이트 단위로 파일에 저장**
  - 첫 번째 인자(ptr): 저장할 데이터의 메모리의 주소
  - 두 번째 인자(size): 저장할 자료 항목의 바이트 크기 (한 항목의 크기)
  - 세 번째 인자(nitems): 출력될 항목의 개수
  - 마지막 인자: 출력될 파일 포인터
  - 파일 stream에 ptr에서 시작해서 (size * nitems) 바이트만큼의 데이터를 파일로 저장
    - 리턴값: 성공적으로 저장된 항목(size)의 개수(nitems 수)

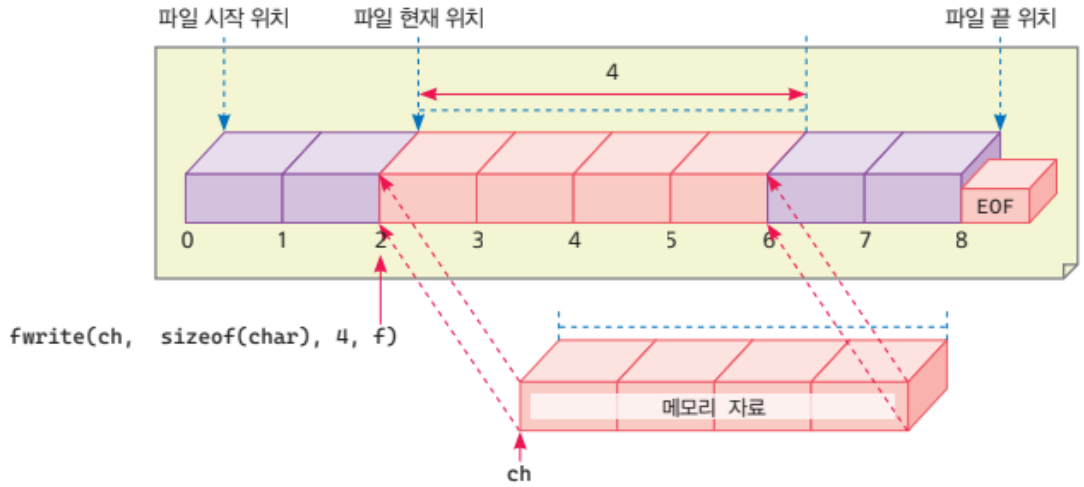
```
size_t fwrite(const void *ptr, size_t size, size_t nitems, FILE *stream);
```

- **함수 fread(): 바이트 단위로 파일을 읽음**
  - 이진 파일에 저장되어 있는 자료를 읽음
  - 파일에서 (size * nitems) 바이트만큼의 자료를 읽어서 ptr에 저장
    - 리턴값: 파일에서 성공적으로 읽은 항목(size)의 개수(nitems 수)
    - 파일의 끝 부분: < nitems
    - 파일의 끝에 도달: 0 리턴

```
size_t fread(const void *ptr, size_t size, size_t nitems, FILE *stream);
```

함수 fwrite() 이용 과정

- **함수 fwrite()**
  - 바이트 단위로 원하는 블록을 파일에 출력하기 위한 함수
  - 출력된 자료는 함수 fread()로 읽어야 그 자료 유형을 유지
  - 세 번째 항목인 출력 항목 수를 4로 지정한 경우의 출력: sizeof(char) * 4 = 4 bytes를 저장



- **이진 파일을 위한 파일 열기 모드**
  - 문자 'b'를 추가

모드	의미
"rb"	이진 파일을 읽기(read) 모드로 연다
"wb"	이진 파일을 쓰기(write) 모드로 연다
"ab"	이진 파일을 추가(append) 모드로 연다

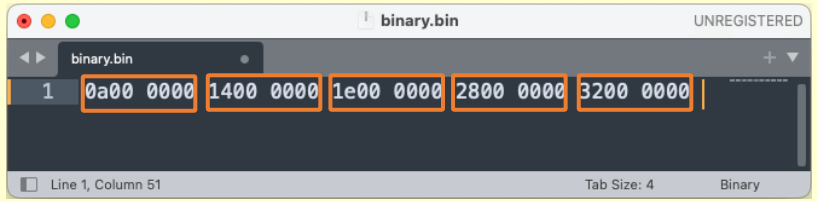
fwrite_fread.c (실습)

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

#define SIZE 5
int fwrite_ex()
{
    int buffer[SIZE] = {10, 20, 30, 40, 50};
    FILE *fp = NULL;
    int count = 0;
    fp = fopen("binary.bin", "wb");

    if (fp == NULL)
    {
        fprintf(stderr, "binary.bin 파일을 열 수 없습니다.\n");
        exit(1);
    }
    // buffer: 메모리 블록 주소,
    // sizeof(int): 한 항목의 크기,
    // count: 항목의 개수, fp: 파일 포인터
    count = sizeof(buffer) / sizeof(int);
    fwrite(buffer, sizeof(int), count, fp);
    fclose(fp);
    return 0;
}
```

실행 결과 10 20 30 40 50

```c
int fread_ex()
{
    int buffer[SIZE];
    FILE *fp = NULL;
    int cnt = 0;

    fp = fopen("binary.bin", "rb");
    if (fp == NULL)
    {
        fprintf(stderr, "binary.bin 파일을 열 수 없습니다.");
        exit(1);
    }
    cnt = fread(buffer, sizeof(int), SIZE, fp);
    for (int i = 0; i < cnt; i++)
        printf("%d ", buffer[i]);
    printf("\n");
    fclose(fp);
    return 0;
}

int main()
{
    fwrite_ex();
    fread_ex();
    return 0;
}
```

cnt: sizeof(int) 크기의 개수

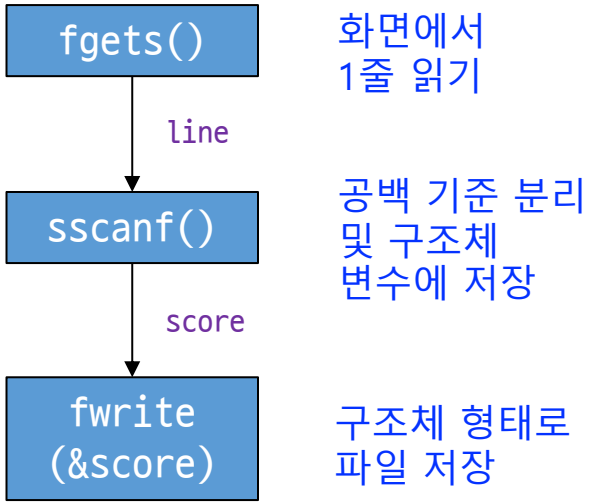


학생 성적 구조체 파일 쓰기

- **구조체 personscore: 학생의 성적정보를 구조체로 표현**
  - 번호, 이름, 중간, 기말, 퀴즈 점수를 멤버로 구성

```
typedef struct personscore    //구조체 struct personscore 정의
{
    int number;               //번호
    char name[40];            //이름
    int mid;                  //중간성적
    int final;                //기말성적
    int quiz;                 //퀴즈성적
} pscore;                     //구조체 자료형 pscore 정의
```

- **표준입력으로 여러 명의 구조체 자료형의 데이터를 입력 받음**
  - 구조체 자료형을 파일 "score.bin"에 저장하는 프로그램
  - 표준 입력은 사람마다 한 행씩 입력 받음
    - ➢ 입력된 학생 수로 학생 번호를 입력
  - 행마다 fgets()를 이용하여 하나의 문자열로 받고
    - ➢ 입력된 문자열에서 각 구조체의 멤버 자료를 추출
    - ➢ 문자열에서 자료를 추출하기 위하여 함수 sscanf()를 이용



구조체를 저장하는 이진 파일: 06structfio.c #1

<06structfio.c>

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

typedef struct personscore // 구조체 struct personscore 정의
{
    int number; // 번호
    char name[40]; // 이름
    int mid; // 중간성적
    int final; // 기말성적
    int quiz; // 퀴즈성적
} pscore; // 구조체 자료형 pscore 정의

void printhead()
{
    printf("%s\n", "-------------------------------------------------");
    printf("%4s %10s %8s %8s %8s\n", "번호", "이름", "중간", "기말", "퀴즈");
    printf("%s\n", "-------------------------------------------------");
}
```

구조체를 저장하는 이진 파일: 06structfio.c #2

```c
int main()
{
    char fname[] = "score.bin";
    FILE *f;

    if ((f = fopen(fname, "wb")) == NULL)
    {
        printf("파일이 열리지 않습니다.\n");
        exit(1);
    }

    char line[80]; // 표준입력으로 행을 저장하기 위한 변수
    int cnt = 0; // 입력 학생 번호(자동 생성) 변수
    pscore score; // 구조체 변수 선언
    printf("이름과 성적(중간, 기말, 퀴즈)을 입력하세요.\n");

    fgets(line, 80, stdin);
    while (!feof(stdin))
    {
        // 표준 입력으로 받은 한 줄의 문자열을 구조체 멤버로 입력
        sscanf(line, "%s %d %d %d", score.name, &score.mid, &score.final, &score.quiz);
        score.number = ++cnt;
        fwrite(&score, sizeof(pscore), 1, f);
        fgets(line, 80, stdin);
    }
    fclose(f);
```

구조체 내용을 파일로 저장
- 저장 단위: psocre 구조체 크기 1개씩 저장

구조체를 저장하는 이진 파일: 06structfio.c #3

```
    if ((f = fopen(fname, "rb")) == NULL)
    {
        printf("파일이 열리지 않습니다.\n");
        exit(1);
    }
    printhead();

    fread(&score, sizeof(pscore), 1, f); // 이진모드로 파일 f에서 구조체 pscore 자료 읽기
    while (!feof(f)) // 파일에서 읽어 표준출력에 쓰기
    {
        fprintf(stdout, "%4d %8s %6d %6d %6d\n",
                        score.number, score.name, score.mid, score.final, score.quiz);
        fread(&score, sizeof(pscore), 1, f);
    }
    printf("%s\n", "-----------------------------------------------");
    fclose(f);

    return 0;
}
```

파일에서 (psocre 구조체 크기* 1개씩) 읽어옴

실행 결과

```
이름과 성적(중간, 기말, 퀴즈)을 입력하세요.
Hong 98 88 78
Kim 78 98 45
Lee 88 85 95
Ctrl+Z 또는 Ctrl+D

-----------------------------------------------
번호     이름   중간   기말   퀴즈
-----------------------------------------------
   1     Hong     98     88     78
   2      Kim     78     98     45
   3      Lee     88     85     95
-----------------------------------------------
```

LAB. 학생 정보 구조체 배열 파일 쓰고 읽기

- **학생 정보를 표현하는 구조체 student**
  - 학과와 이름, 학번으로 구성

```c
//구조체 자료형 student 정의
typedef struct student
{
    char dept[40]; //학과
    char name[20]; //이름
    int snum; //학번
}student;
```

- **학생 3명의 정보를 초기화로 배열 mylab에 저장한 후**
  - 파일 student.bin을 쓰기 모드로 열고 함수 fwrite()로 배열 데이터를 모두 파일로 저장
  - student.bin을 읽기 모드로 열고 함수 fread()로 저장된 데이터를 모두 입력
  - 배열 lab에 저장된 정보를 다시 출력

LAB 소스 (실습)

<lab3stdtinfo.c>

```c
#include <stdio.h>
// 구조체 자료형 student 정의
typedef struct student
{
    char dept[40]; // 학과
    char name[20]; // 이름
    int snum; // 학번
} student;

int main()
{
    student mylab[] = {
            {"컴퓨터정보공학과", "이미정", 20224576},
            {"컴퓨터정보공학과", "김별이", 20226734},
            {"컴퓨터소프트웨어공학과", "김한수", 20238732}};

    FILE *f;
    char fname[] = "student.bin";

    f = fopen(fname, "wb");
    int size = sizeof(mylab) / sizeof(student);
    fwrite(mylab, sizeof(student), size, f);
    fclose(f);

    f = fopen(fname, "rb");
    student lab[size]; // 파일의 내용을 저장할 배열 선언

    // 파일 f에서 student 크기로 size 수만큼 읽어 lab에 저장
    fread(lab, sizeof(student), size, f);
    for (int i = 0; i < size; i++)
        fprintf(stdout, "%24s%10s%12d\n",
                lab[i].dept, lab[i].name, lab[i].snum);

    fclose(f);

    // 파일에서 구조체 하나씩 읽어 읽을 내용이 있으면 출력
    /*
    student one;
    while (fread(&one, sizeof(student), 1, f))
        fprintf(stdout, "%24s%10s%12d\n",
                    one.dept, one.name, one.snum);
     fclose(f);
    */
    return 0;
}
```

student 크기 * size(3) 크기 저장

실행 결과

```
컴퓨터정보공학과 이미정     20224576
컴퓨터정보공학과 김별이     20226734
컴퓨터소프트웨어공학과 김한수     20238732
```

이미지 파일 복사

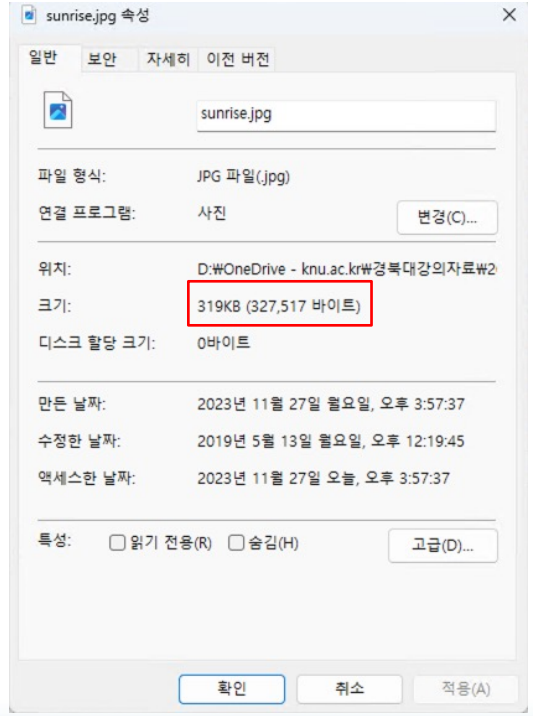
- **fread()함수**
  - char buffer[1024];
  - rd_bytes = fread(buffer, sizeof(char), sizeof(buffer), src_file)
  - rd_bytes: 실제 읽어온 바이트 수 (buffer의 크기[1024]가 아님)
    - ➢ 읽어 올 수 있는 파일의 크기가 buffer의 크기보다 큰 경우
      - – buffer의 크기만큼 읽어옴
    - ➢ 읽어 올 수 있는 파일의 크기가 buffer의 크기보다 작은 경우
      - – 실제 남아 있는 바이트 수만큼 읽어오고 실제 읽은 바이트 수를 리턴함

- **fwrite() 함수**
  - wr_bytes = fwrite(buffer, 1, rd_bytes, dst_file)
  - 파일 저장: 실제 읽어온 바이트 수만큼 파일로 저장

- **결과**
  - sunrise.jpg : 327,517 Bytes

1024	1024	1024	1024	...	861

<sunrise.jpg>



이미지 파일 복사 예제 #1 (실습)

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

int main()
{
    FILE *src_file, *dst_file;
    char srcname[100] = {0};
    char dstname[100] = {0};
    char buffer[1024];
    int rd_byte;
    int total_byte = 0;

    printf("Type the original image file name: ");
    scanf("%s", srcname);

    printf("Type a file name for saving: ");
    scanf("%s", dstname);

    src_file = fopen(srcname, "rb");
    dst_file = fopen(dstname, "wb");
    if (src_file == NULL || dst_file == NULL)
    {
        fprintf(stderr, "File open error \n");
        exit(1);
    }
    while ((rd_bytes = fread(buffer, sizeof(char), sizeof(buffer), src_file)) > 0)
    {
        total_byte += rd_bytes; // 전체 파일 크기 계산
        // 읽어온 바이트 수만큼 파일로 저장함(rd_bytes)
        int wr_bytes = fwrite(buffer, 1, rd_bytes, dst_file);

        printf("rd_bytes= %d, wr_bytes= %d\n", rd_bytes, wr_bytes);

    }
    printf("Complete file copy (%s to %s) File size= %d Bytes\n",
                        srcname, dstname, total_byte);

    fclose(src_file);
    fclose(dst_file);

    return 0;
}
```

```c
rd_bytes = fread(buffer, sizeof(char), sizeof(buffer), src_file);
```

- rd_bytes = 1 byte * sizeof(buffer): 실제 읽은 바이트 수

```c
rd_bytes = fwrite(buffer, 1, rd_bytes, dst_file);
```

- 실제 읽은 바이트 수만큼 dst_file에 저장

이미지 파일 복사 예제 실행 결과

원본: sunrise.jpg

복사본: sunrise_copy.jpg

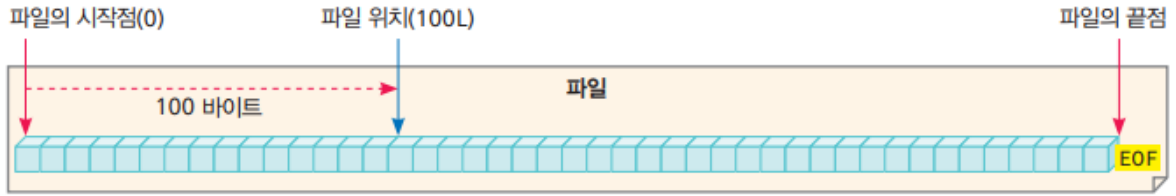
```
Type the original image file name: sunrise.jpg
Type a file name for saving: sunrize_copy.jpg
rd_bytes= 1024, wr_bytes= 1024
rd_bytes= 1024, wr_bytes= 1024
rd_bytes= 1024, wr_bytes= 1024
rd_bytes= 1024, wr_bytes= 1024
. . .

rd_bytes= 1024, wr_bytes= 1024
rd_bytes= 1024, wr_bytes= 1024
rd_bytes= 861, wr_bytes= 861
Complete file copy (sunrise.jpg to sunrize_copy.jpg) File size= 327517 Bytes
```

rd_bytes = 1 byte * sizeof(buffer)

파일의 마지막 부분 남은 바이트 수

파일 위치

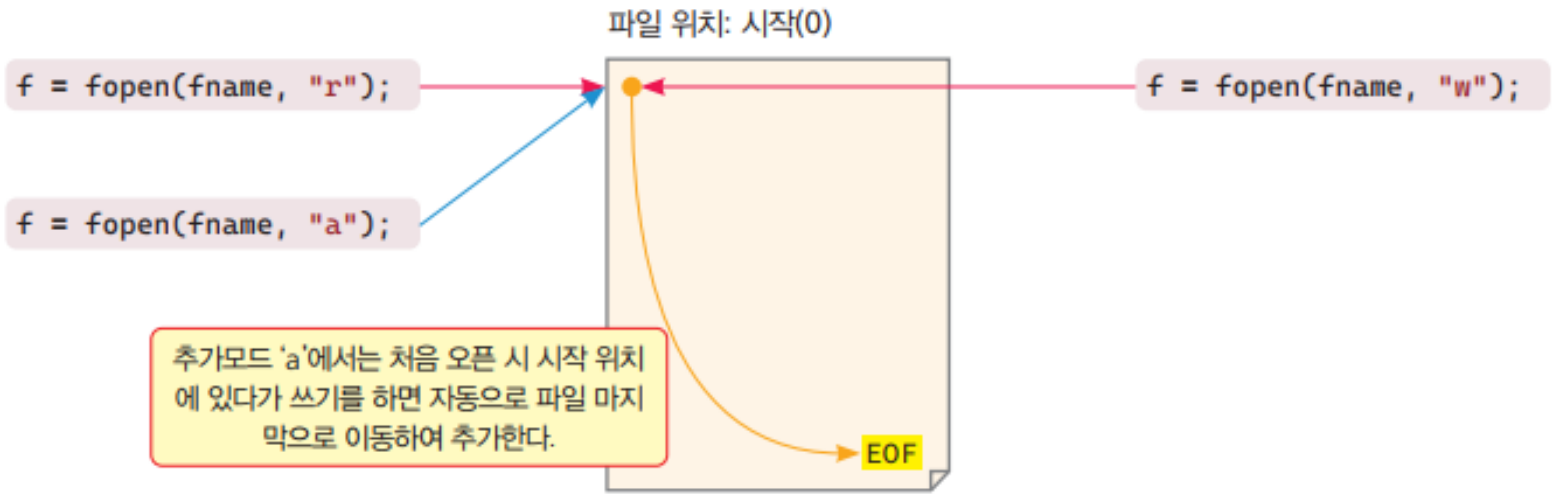
- **바이트 단위로 파일 내부 위치를 나타내는 값**
  - '파일 지시자(file indicator)' 또는 '파일 표시자' 라고 부름
  - 파일의 시작점에서 파일 위치는 0이며 1 바이트마다 1씩 증가
  - 파일의 마지막에는 파일의 마지막임을 알리는 EOF(End Of File) 표시
- **파일을 열면**
  - 파일 위치(file position)는 항상 파일의 시작 부분을 가리킴



- **만일 파일 위치가 100L**
  - 파일의 처음에서부터 100바이트 떨어진 위치에 현재 파일 위치
  - 파일 위치 값은 일반적으로 자료형 long으로 취급
    - ➢ 상수를 기술할 때 100L처럼 수 뒤에 L을 기술
- **파일에 내용을 쓰거나 읽으려면 그 파일 위치로 이동**
  - 자료를 읽거나 쓰는 만큼 파일의 현재 위치에서 뒤로 이동

파일 스트림 연결 시 파일 위치

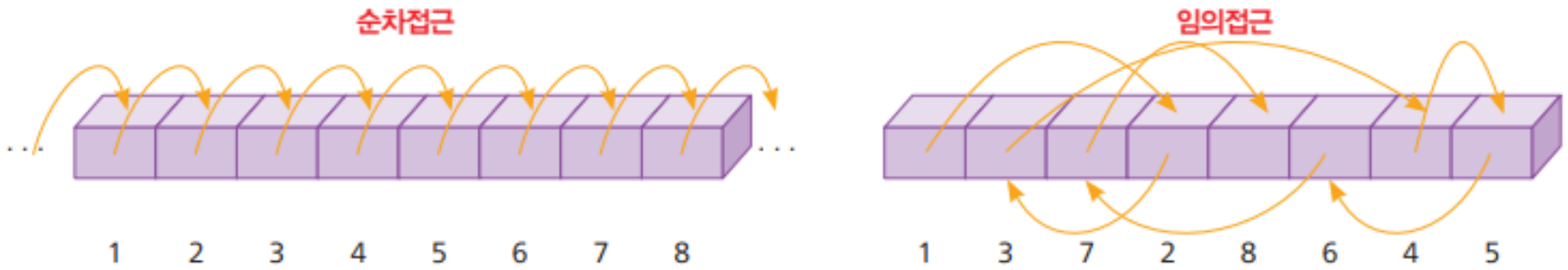
- **파일을 처음으로 열면 모드에 관계없이 파일 위치는 모두 0**
- **파일 모드가 추가(a)**
  - 파일을 처음 열면 파일 위치는 0
    - ➢ 자료를 파일에 쓰면 자동으로 파일 위치가 마지막으로 이동되어 추가
  - 파일 위치를 임의로 이동하였다면 파일의 마지막으로 이동하여 추가



순차, 임의 접근

- **순차적 접근(sequential access)**
  - 파일 위치를 처음부터 하나씩 증가 하면서 파일을 참조하는 방식

- **임의 접근(random access)**
  - 파일의 어느 위치든 바로 참조하는 방식
  - 관련 함수 `fseek()`, `ftell()`, `rewind()` 등을 활용하여 접근



함수 fseek()

- **파일의 임의 접근을 처리**
  - 함수 fseek()이 필요
    - 파일 위치를 자유자재로 이동하는 함수
    - 헤더파일 stdio.h 필요
  - 첫번째 인자: 파일 포인터
  - 두 번째 인자: 오프셋(offset)
    - long 유형으로 기준점으로부터 떨어진 값
  - 세 번째 인자: 오프셋을 계산하는 기준점
    - 정수형 기호 상수로 다음 세 가지 중의 하나

함수 fseek() 함수원형

```
int fseek(FILE * _File, long _Offset, int _Origin);
```

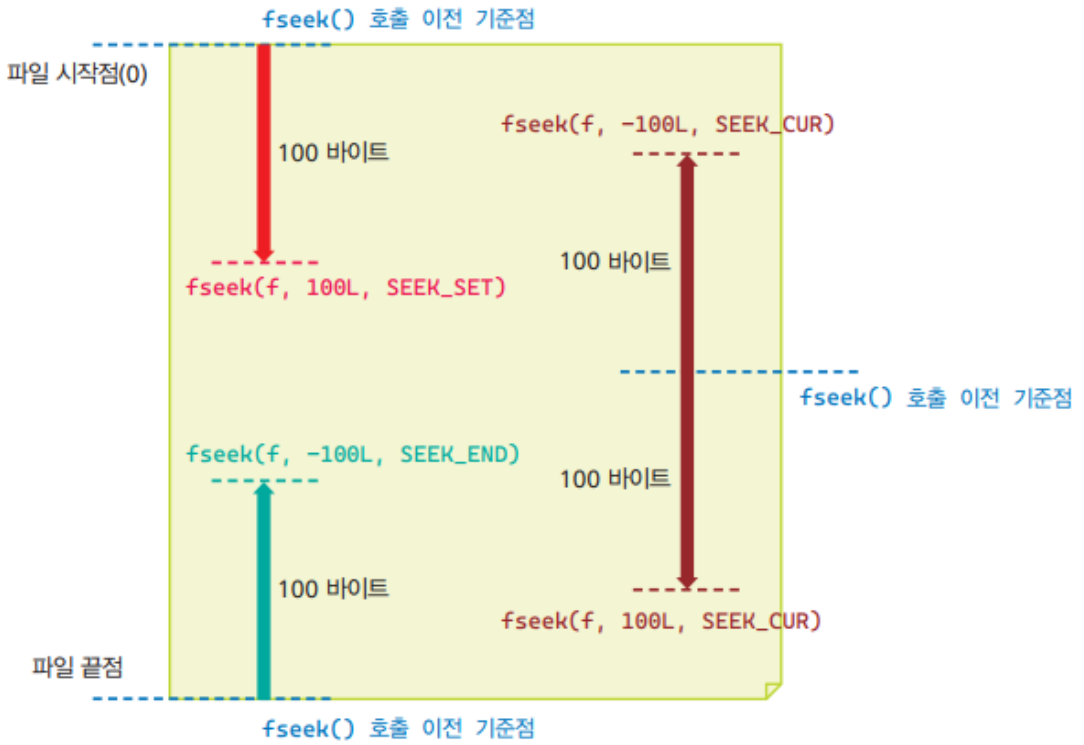
함수 fseek()는 파일 _File의 기준점 _Origin에서 _Offest만큼 파일 포인터를 이동하는 함수 , 성공하면 0을 반환하며 실패하면 0이 아닌 정수를 반환

```
fseek(f, 0L, SEEK_SET);
fseek(f, 100L, SEEK_CUR);
fseek(f, -100L, SEEK_END);
```

기호	값	의미
SEEK_SET	0	파일의 시작 위치
SEEK_CUR	1	파일의 현재 위치
SEEK_END	2	파일의 끝 위치

함수 fseek() (2)

- **함수 fseek(f, 100L, SEEK_SET)의 호출**
  - 파일 위치를 파일의 처음 위치에서 100바이트 떨어진 위치로 이동
- **함수 fseek(f, 100L, SEEK_CUR)의 호출**
  - 파일의 현재 위치에서 100바이트 떨어진 위치로 이동
- **함수 fseek(f, -100L, SEEK_END)의 호출**
  - 파일 끝 위치에서 앞으로 100바이트 떨어진 위치로 이동
- **함수 fseek()에서 offset**
  - 양수이면 파일의 끝점으로,
  - 음수이면 파일의 시작점으로의 이동방향을 표시

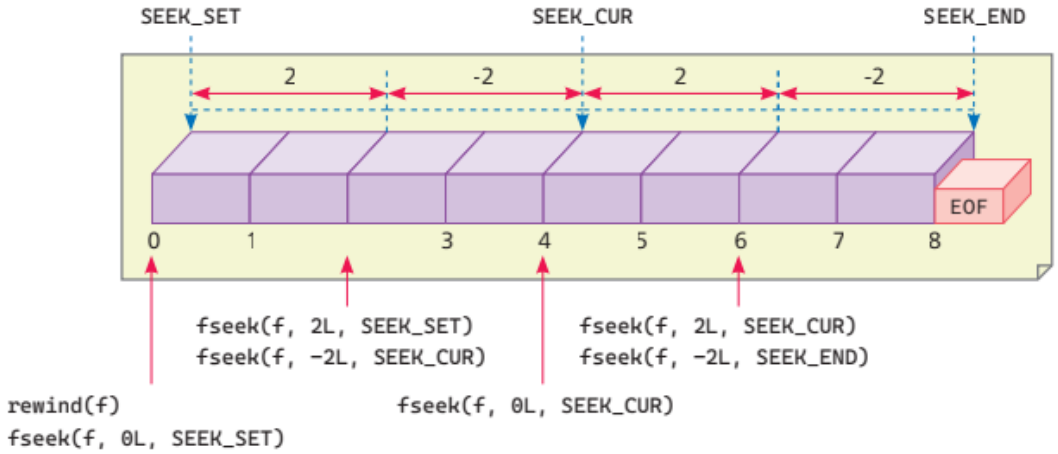


함수 fseek() (3)

- **파일 위치와 관련된 함수**
  - 함수 ftell(): 인자인 파일의 파일 위치를 반환
  - 함수 rewind(): 파일 위치를 무조건 가장 앞으로 이동

함수	기능
int fseek(FILE *, long offset, int pos)	파일 위치를 기준점(pos)으로부터 offset만큼 이동
long ftell(FILE *)	파일의 현재 위치를 반환
void rewind(FILE *)	파일의 현재 위치를 0(파일의 시작점)으로 이동

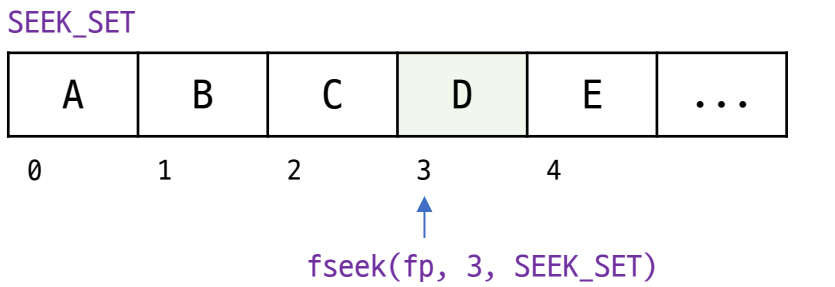
- **함수 rewind() 호출: 파일의 맨 처음으로 이동**
  - 함수 fseek(f, 0L, SEEK_SET)와 동일한 기능 수행

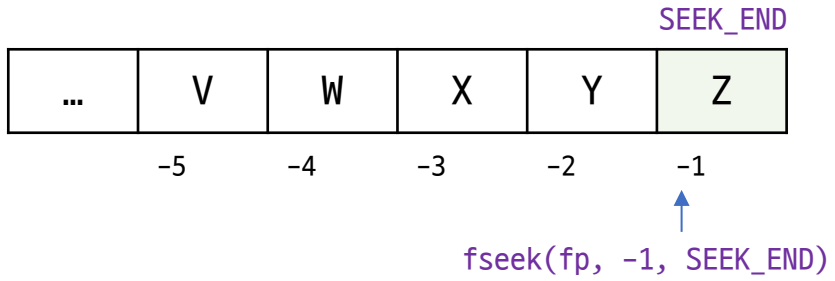


fseek 예제

<fseek_ex.c>

```c
#include <stdio.h>

int main(void)
{
    FILE *fp;

    fp = fopen("sample.txt", "w");
    fputs("ABCDEFGHIJKLMNOPQRSTUVWXYZ", fp);
    fclose(fp);

    fp = fopen("sample.txt", "r");
    fseek(fp, 3, SEEK_SET);
    printf("fseek(fp, 3, SEEK_SET)= %c \n", fgetc(fp));

    fseek(fp, -1, SEEK_END);
    printf("fseek(fp, -1, SEEK_END)= %c \n", fgetc(fp));

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





실행 결과

```
fseek(fp, 3, SEEK_SET)= D
fseek(fp, -1, SEEK_END)= Z
```

파일 열기 다양한 모드 #1

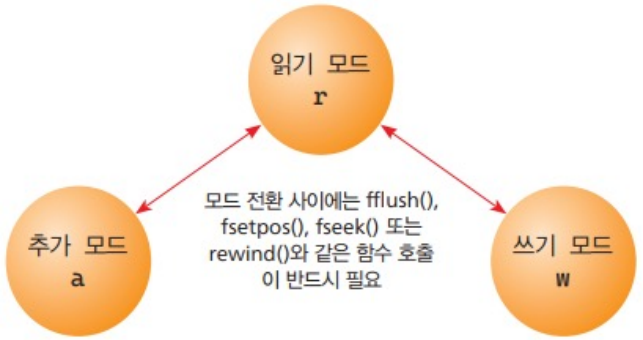
- **함수 fopen()의 파일 열기 모드 종류**
  - 텍스트 파일인 경우 "r", "w", "a", "r+", "w+", "a+" 등

모드	의미			
	파일 열기 모드	모드 전환	파일이 있는 경우	파일이 없는 경우
r	읽기(read)	쓰기(write) 불가능	파일의 처음에서 읽기 시작	에러 발생
w	쓰기(write)	읽기(read) 불가능	이전 내용을 지워지고 파일의 처음부터 쓰기 시작	새로 생성
a	추가(append)	읽기(read) 불가능	파일의 마지막에서 파일의 쓰기 시작하며, 파일 중간에 쓰는 것은 불가능	새로 생성
r+	읽기(read)	쓰기(write)	파일의 처음에서 읽기 시작	에러 발생
w+	쓰기(write)	읽기(read)	이전 내용을 지워지고 파일의 처음부터 쓰기 시작	새로 생성
a+	추가(append)	읽기(read)	파일의 마지막에서 파일의 쓰기 시작하며, 파일 중간에 쓰는 것은 불가능	새로 생성

- **파일모드**
  - 읽기모드 r: 읽기만 가능한 모드이며, 쓰기는 불가능
  - 쓰기모드 w: 파일 어디에든 쓰기가 가능한 모드이나 읽기는 불가능
  - 추가모드 a
    - 파일 중간에 쓸 수 없으며 파일 마지막에 추가적으로 쓰는 것만 가능한 모드
    - 읽기는 불가능
    - 파일에 쓰는 내용은 무조건 파일 마지막에 추가

파일 열기 다양한 모드 #2

- **파일 모드에서 +의 삽입**
  - 수정(update) 모드 의미
    - 원래의 모드에서 읽기 또는 쓰기가 추가되는 모드
    - 모드 간의 전환이 가능
- **파일 모드 r+**
  - 처음에 읽기 모드로 파일을 열어
    - 쓰기 모드로 전환 가능
  - 파일이 없으면 오류가 발생
- **파일 모드 w+**
  - 처음에 쓰기 모드로 파일을 열어
    - 필요하면 읽기 모드로 전환 가능
  - 만일 파일이 존재한다면
    - 이전의 내용은 모두 사라짐



- **파일 모드 a+**
  - 처음에 추가 모드로 파일을 열어
    - 필요하면 읽기 모드로 전환 가능
- **수정 모드에서 모드 전환**
  - 추가 모드와 읽기모드, 쓰기 모드와 읽기 모드 사이의 전환이 가능
  - 파일 모드 전환 사이에는 fflush()와 fseek() 또는 rewind()와 같은 함수 호출이 필요
- **이진 파일을 위한 파일 열기 모드**
  - 문자 'b'를 추가 가능

모드		의미
rb		이진 파일의 읽기(read) 모드로 파일을 연다.
wb		이진 파일의 쓰기(write) 모드로 파일을 연다.
ab		이진 파일의 추가(append) 모드로 파일을 연다.
rb+	r+b	이진 파일의 읽기(read)와 쓰기(write) 모드로 파일을 연다.
wb+	w+b	이진 파일의 읽기(read)와 쓰기(write) 모드로 파일을 연다.
ab+	a+b	이진 파일의 추가(append) 모드로 파일을 연다.

학생의 성적 정보를 추가하는 프로그램

- **제일 먼저 파일 score.bin 파일에 있는 학생 정보를 모두 읽어와 출력**
  - 06structfio.c 에서 저장한 score.bin 파일 사용
  - 파일에 있는 마지막 학생 정보로부터 마지막 학생 번호를 알아냄
  - 이 번호에 1씩 증가시키면서 다음에 추가될 학생의 번호로 이용
  - 추가될 학생 정보는 학생마다 한 행씩 자료를 받아서 파일 score.bin 파일에 추가

- **키보드 ctrl + z를 누르면 입력이 종료**
  - 다시 파일 score.bin에서 모든 자료를 읽어 모든 정보를 출력
  - score.bin 파일
    - "ab+"로 열기
    - 학생 정보를 추가도 하고, 다시 읽기도 하기 위함

실습 예제 15-7 학생 성적 구조체 추가 프로그램 #1

<07appendsfile.c>

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

struct personscore
{
    int number;
    char name[40];
    int mid;
    int final;
   int quiz;
};

typedef struct personscore pscore;

void printhead();
int printscore(FILE* f);
void appendscore(FILE* f, int cnt);
```

실습 예제 15-7 학생 성적 구조체 추가 프로그램 #2

<07appendsfile.c>

```c
int main()
{
    FILE* f;
    char fname[] = "score.bin";
    int cnt = 0;
    long offset = 0;

    if ((f = fopen(fname, "ab+")) == NULL)
    {
        printf("파일이 열리지 않습니다.\n");
        exit(1);
    }

    int readcnt = printscore(f);
    if (readcnt == 1)
    {
        pscore score;
        offset = (long)sizeof(pscore); //구조체 하나의 크기
        //파일의 마지막에서 마지막 학생을 읽기 위해 뒤로 이동
        fseek(f, -offset, SEEK_END);
        fread(&score, sizeof(pscore), 1, f); //마지막 학생을 읽음
        cnt = score.number; //제일 마지막 자료의 번호
        printf("\n제일 마지막 번호가 %d번 입니다. \n\n", cnt);
    }
    fseek(f, 0L, SEEK_END);
    appendscore(f, cnt);
    printscore(f);
    fclose(f);

    return 0;
}

void appendscore(FILE* f, int cnt)
{
    char line[80];
    pscore score = { 0 };
    printf("추가할 이름과 성적(중간, 기말, 퀴즈)을 입력하세요.\n\n");
    fgets(line, 80, stdin);
    while (!feof(stdin))
    {
        sscanf(line, "%s %d %d %d", score.name, &score.mid,
                                    &score.final, &score.quiz);
        score.number = ++cnt;
        fwrite(&score, sizeof(pscore), 1, f);
        fgets(line, 80, stdin);
    }
}
```

실습 예제 15-7 학생 성적 구조체 추가 프로그램 #3

<07appendsfile.c>

```
int printscore(FILE* f)
{
    rewind(f); //파일의 맨 앞으로 이동
    pscore score;
    int readcnt = fread(&score, sizeof(pscore), 1, f);
    if (readcnt == 0) //파일 f에 하나도 자료가 없으면 변수 readcnt가 0
    {
        printf("현재는 성적 정보가 하나도 없습니다. >>\n");
        return 0;
    }
    printhead(); //제목 출력
    while (!feof(f)) //표준출력에 쓰기
    {
        fprintf(stdout, "%6d%18s%8d%8d%8d\n",
                score.number, score.name, score.mid, score.final, score.quiz);
        fread(&score, sizeof(pscore), 1, f);
    }
    fprintf(stdout, "%s\n", " ------------------------------------------------");

    return 1;
}

void printhead()
{
    printf("\n현재의 성적 내용은 >>\n");
    fprintf(stdout, "%s\n", " ------------------------------------------------");
    fprintf(stdout, "%8s%15s%10s%8s%8s\n", "번호", "이름", "중간", "기말", "퀴즈");
    fprintf(stdout, "%s\n", " ------------------------------------------------");
}
```

실습 예제 15-7 학생 성적 구조체 추가 프로그램 #4

실행 결과

```
현재의 성적 내용은 >>
  ---------------------------------------------
  번호         이름    중간  기말  퀴즈
  ---------------------------------------------
     1             Hong     98     88     78
     2              Kim     78     98     45
     3              Lee     88     85     95
  ---------------------------------------------

제일 마지막 번호가 3번 입니다.

추가할 이름과 성적(중간, 기말, 퀴즈)을 입력하세요.
Park 55 69 90
^Z 또는 ^D
현재의 성적 내용은 >>
  ---------------------------------------------
  번호         이름    중간  기말  퀴즈
  ---------------------------------------------
     1             Hong     98     88     78
     2              Kim     78     98     45
     3              Lee     88     85     95
     4             Park     55     69     90
  ---------------------------------------------
```

입출력 함수: 텍스트 파일

- **표준 입출력 장치를 이용한 입출력 함수**
  - 대부분 헤더파일 stdio.h 필요

기호	종류	표준 입출력	파일 입출력
문자	입력	int getchar(void)	int fgetc(FILE *), int getc(FILE *)
	출력	int putchar(int)	int fputc(int, FILE *), int putc(int, FILE *)
문자열	입력	char *gets(char *)	char *fgets(char *, int, FILE *)
	출력	int puts(const char *)	int fputs(const char *, FILE *)
서식 자료	입력	int scanf(const char *, ...)	int fscanf(FILE *, const char *, ...)
	출력	int printf(const char *, ...)	int fprintf(FILE *, const char *, ...)

입출력 함수: 이진 파일

- **블록과 정수 자료의 파일 입출력 함수**
  - 함수 getw()와 putw()
    - 워드(word)크기의 int형 정수를 파일에 이진 모드로 입출력 하는 함수
    - 헤더파일 stdio.h 필요

기호	종류	파일 입출력
블록	입력	size_t fread(void *, size_t, size_t, FILE *)
	출력	size_t fwrite(const void *, size_t, size_t, FILE *)
정수(int)	입력 (Windows)	int _getw(FILE *)
	출력 (Windows)	int _putw(int, FILE *)

파일 처리 함수 remove()와 rename()

- **함수 remove(): 지정된 특정한 파일을 삭제**
  - remove("sample.txt")
    - ➢ 문자열로 지정된 파일을 삭제
  - 파일이 성공적으로 삭제되면 0을 반환하며, 실패하면 -1을 반환

- **함수 rename(): 지정된 파일 또는 폴더의 이름을 새로운 이름으로 바꾸는 기능을 수행**
  - rename("oldname.txt", "newname.txt")
    - ➢ 앞의 파일이름 또는 폴더이름을 뒤에 지정한 이름으로 바꾸는 역할
  - 성공적으로 이름이 수정되면 0을 반환하며, 실패하면 0이 아닌 정수를 반환

- **헤더 파일 stdio.h에 그 함수 원형이 정의**

기능	함수 원형
파일 삭제	int remove(const char *filename)
파일 또는 폴더 이름 변경	int remane(const char *old, const char *new)

파일 이름 수정

- **함수 rename()을 이용한 예제 프로그램**
  - 프로젝트 폴더에서 파일 old.c가 new.c로 변경된 것을 확인

<08frename.c>

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

int main()
{
    char *fname1 = "old.c";
    char *fname2 = "new.c";
    int result = rename(fname1, fname2); // 파일 이름 수정 함수 호출

   if(result == 0)
        printf("파일 %s가 %s로 수정되었습니다.\n", fname1, fname2);
    else
        printf("파일 이름 변경 오류\n");

    return 0;
}
```

```
rename result: 0
파일 old.c가 new.c로 수정되었습니다.
```

<old.c>

chap15 > C old.c > ...

```c
#include <stdio.h>

int main()
{
    printf("Old.c\n");
    return 0;
}
```

<new.c>

chap15 > C new.c > ...

```c
#include <stdio.h>

int main()
{
    printf("Old.c\n");
    return 0;
}
```

LAB 정수 배열값을 파일로 저장 후 다시 파일 읽기

- **1차원 정수 배열의 내용을 모두 파일에 출력한 후**
  - 다시 파일에서 입력으로 받아 표준 출력
- **방법**
  - 파일은 "wb+"로 열어 이진 모드로 쓰고 읽을 수 있도록
  - 배열의 내용을 파일 "test.bin"에 모두 출력
  - 파일에 출력 후, 다시 처음부터 읽으려면 함수 rewind()를 호출
  - 파일 "test.bin"에서 정수를 읽을 때는 한 정수 읽은 후 함수 fseek()을 사용하여 파일 위치를 하나 건너 뛰고 다음 정수를 읽는 식으로 입력하여 다시 표준출력
- **결과**
  - 파일에 출력 자료: 10 20 30 40 50 60 70 80
  - 파일에서 입력 자료(하나씩 건너 뜀): 10 30 50 70

LAB 정수 배열값을 파일에 저장 후 다시 파일 읽기 예제

<lab4arrayput.c>

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

int main()
{
    FILE *f;
    int value = 0;

    if ((f = fopen("test.bin", "wb+")) == NULL)
    {
        printf("파일이 열리지 않습니다.\n");
        exit(1);
    }

    int out[] = {10, 20, 30, 40, 50, 60, 70, 80};
    int size = sizeof(out) / sizeof(out[0]);
    printf("파일에 출력 자료: ");
    for (int i = 0; i < size; i++)
    {
        fwrite(&out[i], sizeof(int), 1, f);
        printf("%d ", out[i]);
    }
    printf("\n");
    rewind(f); // 파일 f의 파일 지시자를 0으로 위치

    printf("파일에서 입력 자료(하나씩 건너 뜀): ");
    for (int i = 0; i < size / 2; i++)
    {
        fread(&value, sizeof(int), 1, f);
        printf("%d ", value);

        // 다음 정수는 하나 건너 뛰기
        fseek(f, sizeof(int), SEEK_CUR);
    }

    printf("\n");
    fclose(f);

    return 0;
}
```

"wb+": 쓰기와 읽기 모드

파일 쓰기 먼저 수행

rewind() 이후 파일 읽기 수행

실행 결과

```
파일에 출력 자료: 10 20 30 40 50 60 70 80
파일에서 입력 자료(하나씩 건너 뜀): 10 30 50 70
```

Questions?