

# 파일생성 ( “w” )

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

int main(void) {
    char filename[] = "test.txt";
    FILE *fp = NULL;
    int i;

    fopen_s(&fp, filename, "w");
    if (fp == NULL) {
        printf("Can't open %s.\n\n", filename);
        return 0;
    }
    for (i = 0; i < 5; i++) {
        fprintf(fp, "문자열\n");
    }
    fclose(fp);
    return 0;
}
```

[test.txt 파일 내용]  
문자열  
문자열  
문자열  
문자열  
문자열

# 파일내용 덧붙이기 ( “a” )

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

File position : 열었을 때는 맨 앞에 위치하나 write하면 맨 뒤로 가서 쓰게 됨.

```
int main(void) {
    char filename[] = "test.txt";
    FILE *fp = NULL;

    fopen_s(&fp, filename, "a");
    printf("start : file pos=%ld\n", ftell(fp));
    if (fp == NULL) {
        printf("Can't open %s.\n\n", filename);
        return 0;
    }
    fprintf(fp, "덧붙인 내용\n");
    printf("end : file pos=%ld\n", ftell(fp));
    fclose(fp);
    return 0;
}
```

[실행결과]  
start : file pos=0  
end : file pos=53

[test.txt 파일 내용]  
문자열  
문자열  
문자열  
문자열  
문자열  
덧붙인 내용

# 파일내용 덧쓰기 ( “r+” )

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

```
int main(void) {
    char filename[] = "test.txt";
    FILE *fp = NULL;

    fopen_s(&fp, filename, "r+");
    printf("start : file pos=%ld\n", ftell(fp));
    if (fp == NULL) {
        printf("Can't open %s.\n\n", filename);
        return 0;
    }
    fprintf(fp, "덧쓰기\n");
    printf("end : file pos=%ld\n", ftell(fp));
    fclose(fp);
    return 0;
}
```

File position : 열었을 때 맨 앞에 위치하고 write하면 그 자리에 쓴다.

[실행결과]  
start : file pos=0  
end : file pos=8

[test.txt 파일 내용]  
덧쓰기  
문자열  
문자열  
문자열  
문자열  
덧붙인 내용

# 파일 새로 만들기 ( “w” )

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

```
int main(void) {
    char filename[] = "test.txt";
    FILE *fp = NULL;

    fopen_s(&fp, filename, "w");
    printf("start : file pos=%ld\n", ftell(fp));
    if (fp == NULL) {
        printf("Can't open %s.\n\n", filename);
        return 0;
    }
    fprintf(fp, "새로 만들기\n");
    printf("end : file pos=%ld\n", ftell(fp));
    fclose(fp);
    return 0;
}
```

File position : 열었을 때 맨 앞에 위치하고 write하면 그 자리에 쓴다.

[실행결과]  
start : file pos=0  
end : file pos=13

[test.txt 파일 내용]  
새로 만들기

# 있으면 덧붙이고 없으면 생성하여 쓰기

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

int main(void) {
    char filename[] = "test.txt";
    FILE *fp = NULL;

    fopen_s(&fp, filename, "w");
    printf("start : file pos=%ld\n", ftell(fp));
    if (fp == NULL) {
        printf("Can't open %s.\n\n", filename);
        return 0;
    }
    fprintf(fp, "새로 만들기\n");
    printf("end : file pos=%ld\n", ftell(fp));
    fclose(fp);
    return 0;
}
```

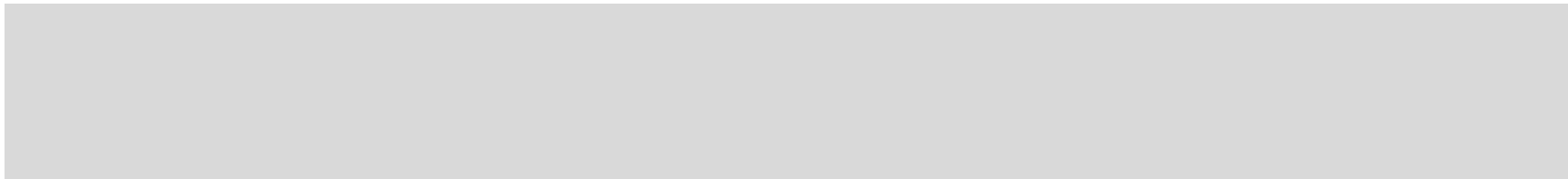
test.txt 파일삭제  
하고 재실행

[실행결과1]  
파일 있음  
start : file pos=0  
end : file pos=52

[test.txt 파일 내용]  
새로 만들기  
있으면 덧붙이고 없으면

[실행결과2]  
파일 없음  
start : file pos=0  
end : file pos=39

[test.txt 파일 내용]  
있으면 덧붙이고 없으면



# 파일

- ◆ **파일의 필요성**
  - 프로그램이 종료되더라도 디스크에 파일로 계속 남길 필요가 있음
- ◆ **파일**
  - 보조기억장치의 정보저장 단위.
- ◆ **파일의 두 가지 유형**
  - **텍스트 파일**
    - ❖ 내용이 아스키 코드(ascii code)와 같은 문자 코드값으로 저장됨
    - ❖ 텍스트 편집기를 통하여 그 내용을 보고 수정 가능
  - **이진 파일**
    - ❖ 실행파일, 그림 파일, 음악 파일, 동영상 파일 등 이진 형태(binary format)로 저장되는 파일
    - ❖ 내용을 이미 알고 있는 특정한 프로그램에 의해 인지될 때 의미가 있음
- ◆ **입출력 스트림(io stream) : 자료의 이동 통로**
  - 표준입력 스트림: 키보드. scanf() 함수 이용
  - 그 외 입력스트림 : 파일입력, 스크린입력(터치스크린), 네트워크입력
  - 표준출력 스트림: 모니터 콘솔. printf() 함수 이용
  - 그 외 출력스트림 : 파일출력, 프린터출력, 네트워크출력

# 파일 스트림

- ◆ 파일 스트림
  - 보조기억장치의 파일과 프로그램을 연결하는 전송경로
- ◆ 파일 스트림 생성
  - 입력/출력 종류, 파일 이름, 파일 모드를 지정하여 생성.
- ◆ 파일 스트림 열기 : 함수 **fopen()** 또는 **fopen\_s()** 이용
  - 특정 파일과 파일 스트림을 연결하는 함수
  - `#include <stdio.h>`
  - FILE 구조체 사용 : FILE \* 타입을 반환
  - 인다 : 파일이름, 파일열기 모드 ("r", "w", "a" 등)
  - 반환 : 성공하면 파일 포인터, 실패하면 NULL
- ◆ 파일 스트림 닫기 : 함수 **fclose()** 이용
  - fopen()으로 연결한 파일 스트림을 닫는 기능을 수행
  - 내부적으로 파일 스트림 연결에 할당된 자원을 반납하고, 파일과 메모리 사이에 있던 버퍼의 내용을 모두 지우는 역할
  - 반환 : 성공하면 0, 실패하면 EOF



# 파일 스트림 사용 (fopen.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h> //for exit()
int main() {
    char *fname = "basic.txt";
    FILE *f;
    char name[30] = "강미정";
    int point = 99;
    if (( f = fopen(fname, "w") ) == NULL ) {
        //if (fopen_s(&f, fname, "w") != 0)
        printf("파일이 열리지 않습니다.\n"); exit(1); };
    fprintf( f, "이름이 %s인 학생의 성적은 %d 입니다.\n", name, point );
    fclose( f );
    printf("이름이 %s인 학생의 성적은 %d 입니다.\n", name, point);
    puts("프로젝트 폴더에서 파일 basic.txt를 메모장으로 열어 보세요.");
    return 0;
}
```

← 파일이름과 파일모드를 주고 열기.

[결과]

이름이 강미정인 학생의 성적은 99 입니다.

프로젝트 폴더에서 파일 basic.txt를 메모장으로 열어 보세요.

# 텍스트 파일 입출력

## (파일에 서식화된 문자열 입출력)

- ◆ **파일에 서식화된 문자열 입출력 함수**
  - `int fprintf( FILE *f, const char *format, ...)`
  - `int fscanf( FILE *f, const char *format, ...)`
  - `int fscanf_s( FILE *f, const char *format, ...)`
  
- ◆ **FILE \* 로서 표준 입출력 표현**
  - `stdin` : 표준입력 (키보드)
  - `stdout` : 표준출력 (모니터)
  - `stderr` : 표준에러 (모니터)

# 텍스트 파일 입출력 (fprintf.c 1/2)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
int main() {
    char fname[] = "grade.txt";
    FILE *f;
    char name[30];
    int point1, point2, cnt = 0;
```

[생성된 파일 내용]  
1 홍길동 90 95

```
if (fopen_s(&f, fname, "w") != 0) {
    printf("파일이 열리지 않습니다.\n");
    exit(1);
};
printf("이름과 성적(중간, 기말)을 입력하세요.\n");
scanf("%s %d %d", name, &point1, &point2);
fprintf(f, "%d %s %d %d\n", ++cnt, name, point1, point2);
fclose(f);
```

파일 "grade.txt"에 쓰기

# 텍스트 파일 입출력

## (fprintf.c 2/2)

```
if ( (f = fopen(fname, "r")) == NULL ) {  
    printf("파일이 열리지 않습니다.\n");  
    exit(1);  
};  
fscanf(f, "%d %s %d %d\n", &cnt, name, &point1, &point2);  
  
fprintf(stdout, "\n%6s%16s%10s%8s\n",  
    "번호", "이름", "중간", "기말");  
fprintf(stdout, "%5d%18s%8d%8d\n",  
    cnt, name, point1, point2);  
fclose(f);  
  
return 0;  
}
```

파일 "grade.txt"에서 읽기

표준출력에 쓰기

[결과]  
이름과 성적(중간, 기말)을 입력하세요.  
홍길동 90 95

번호	이름	중간	기말
1	홍길동	90	95

# 텍스트 파일 입출력

## (파일 문자열 입출력)

- ◆ **char \*fgets( char \*buf, int maxcount, FILE \*fp );**
  - 파일로부터 한 행의 문자열을 입력 받는 함수
  - 공백문자까지 모두 읽고, 개행문자(\n) 만나면 입력 마칩
  - 마지막 개행문자를 '\0' 문자로 바꾸어 입력 버퍼에 저장
  - 첫 번째 인자는 문자열이 저장될 문자 포인터
  - 두 번째 인자는 입력할 문자의 최대 수
  - 세 번째 인자는 입력 문자열을 읽어 올 파일
- ◆ **int fputs( char \*buf, FILE \*fp );**
  - 파일로 한 행의 문자열을 출력하는 함수
  - 첫 번째 인자는 출력될 문자열
  - 두 번째 인자는 문자열이 출력되는 파일
- ◆ **File stream의 상태 체크 함수**
  - **feof()** : File pointer가 end of file에 도달했을 때 1을 반환
  - **ferror()** : 파일 처리에서 오류가 발생했을 때 1을 반환

# 이름, 성적 여러 벌을 입력 받아 파일에 저장. (mlineio.c 1/2)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
int main() {
    char fname[] = "grade.txt";
    FILE *f;
    char names[80];
    int cnt = 0;
    if (fopen_s(&f, fname, "w") != 0) { printf("파일오류!\n"); exit(1);
};
    printf("이름과 성적(중간, 기말)을 입력하세요.\n");
    fgets(names, 80, stdin);
    while ( !feof(stdin) ) {
        fprintf(f, "%d ", ++cnt);
        fputs(names, f);
        fgets(names, 80, stdin);
    }
    fclose(f);
    return 0;
}
```

[결과]

이름과 성적(중간, 기말)을 입력하세요

김철수 59 99

이미영 80 85

홍길동 90 80

^Z

콘솔에 이름 중간 기말 입력하고 Enter 키  
여러 줄에 입력하다가  
종료하고 싶을 때 새줄 첫 행에서 ctrl + Z 누름

Grade.txt 파일에 일련번  
호와 입력 내용 쓰기.

# 텍스트 파일 입출력

## (파일 문자 입출력)

### ◆ 파일로부터 문자 하나를 입력 받는 함수

- `int fgetc( FILE *fp );`
- `int getc( FILE *fp );`
- `int getchar(void);`
  - ❖ 표준 입력에서 읽음.
- `int getche(void);` -> visual C에서는 `_getche`
  - ❖ 표준 입력에서 읽음.

### ◆ 파일로 문자 하나를 출력하는 함수

- `int fputc( int ch, FILE *fp );`
- `int putc( int ch, FILE *fp );`
- `int putchar( int ch );`
  - ❖ 표준 출력에 씀.
- `int putch( int ch );` -> visual C에서는 `_putch`
  - ❖ 표준 출력에 씀.

# 텍스트 파일 입출력 (fget.c)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main() {
    char fname[] = "char.txt"; //입력한 내용이 저장될 파일이름
    FILE *f; //파일 포인터
    if ( (f = fopen(fname, "w")) == NULL ) { printf("파일오류.\n");
exit(1); };
    puts("문자를 입력하다가 종료하려면 x를 입력 >>");
    int ch; //입력된 문자 저장
    while ((ch = _getche()) != 'x')
        fputc(ch, f); //파일에 문자 출력
    fclose(f); puts("");
    if (fopen_s(&f, fname, "r") != 0) { printf("파일오류.\n"); exit(1); };
    while ((ch = fgetc(f)) != EOF)
        _putch(ch); //파일로부터 입력 받은 문자를 표준출력
    fclose(f); puts("");
}
```

[결과]

문자를 입력하다가 종료하려면 x를 입력 >>

abcx

abc

파일 "char.txt"에 쓰기

파일 "char.txt"에서 읽기



# 파일 내용을 표준출력으로 그대로 출력 (list.c)

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    FILE *f;
    if (argc != 2) { printf("사용법: list filename\n"); exit(1); }
    if ( (f = fopen(argv[1], "r")) == NULL ) { printf("파일오류.\n"); exit(1); }

    int ch, cnt = 0;
    printf("%4d: ", ++cnt);
    while ((ch = fgetc(f)) != EOF) {
        putchar(ch); //putc(ch, stdout);
        if (ch == '\n') printf("%4d: ", ++cnt);
    }
    printf("\n");
    fclose(f);
    return 0;
}
```

인자로 받은 파일을 읽기전용으로  
열어서 내용 읽기.

읽은 내용을 출력  
(줄 번호 포함)

[결과]

```
> test.exe grade.txt
1: 1 김철수 59 99
2: 2 이미영 80 85
3: 3 홍길동 90 80
4:
```

# 텍스트 파일과 이진 파일의 입출력

- ◆ 텍스트 모드로 처리 (텍스트 편집기로 내용확인 가능) (파일 열기 모드 't' )
  - `fprintf( FILE *fp, const char *buf, ... );`
  - `fscanf( FILE *fp, const char *buf, ... );`
- ◆ 이진 모드로 처리 (파일 열기 모드 'b' )
  - `fwrite( const void *ptr, size_t size, size_t n, FILE *fp );`
  - `fread( void *destbuf, size_t size, size_t n, FILE *fp );`
- ◆ 구조체 내용 그대로 파일에 쓰기
  - 이진 모드로 처리

# 파일에서 학생 구조체 정보 읽기 (fwrite.c 1/2)

```
#define _CRT_SECURE_NO_WARNINGS
#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;
```

이름과 성적(중간, 기말, 퀴즈)을 입력하세요.

```
name1 90 95 10
name2 80 89 9
name3 90 91 9
^Z
```

# 학생 구조체 정보를 반복 입력받아 파일에 저장

```
int main() {
```

(fwrite.c 2/2)

```
char fname[] = "score.bin";
```

```
FILE *f;
```

```
char line[80];
```

```
int cnt = 0;
```

```
pscore score;
```

쓰기 모드로 파일 열기

```
if ( ( f = fopen(fname, "wb") ) == NULL )
```

```
{ printf("파일이 열리지 않습니다.\n"); exit(1); }
```

```
printf("이름과 성적(중간, 기말, 퀴즈)을 입력하세요.\n");
```

```
fgets(line, 80, stdin);
```

Control+Z 누를 때까지 줄 단위로 입력 받기

```
while (!feof(stdin)) {
```

```
    sscanf(line, "%s %d %d %d",
```

구조체 내용을 파일에 저장

```
        score.name, &score.mid, &score.final, &score.quiz);
```

```
    score.number = ++cnt;
```

```
    fwrite(&score, sizeof(pscore), 1, f);
```

```
    fgets(line, 80, stdin);
```

```
}
```

```
fclose(f);
```

```
return 0;
```

이름과 성적(중간, 기말, 퀴즈)

```
name1 90 95 10
```

```
name2 80 89 9
```

```
name3 90 91 9
```

```
^Z
```

# 파일에서 학생 구조체 정보 읽기 (fread.c 1/2)

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#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;
```

이 부분은 fwrite.c 와 같음

[결과]

번호	이름	중간	기말	퀴즈
1	name1	90	95	10
2	name2	80	89	9
3	name3	90	91	9

# 파일에서 학생 구조체 정보 읽기

## (fread.c 2/2)

```
void printhead() {
    printf("%s\n", " _____");
    printf("%8s%15s%10s%8s%8s\n", "번호", "이름", "중간", "기말", "퀴즈");
    printf("%s\n", " -----");
}

int main() {
    char fname[] = "score.bin";
    FILE *f;
    pscore score;
    if ((f = fopen(fname, "rb")) == NULL) {
        printf("파일이 열리지 않습니다.\n"); exit(1);
    }
    printhead();
    fread(&score, sizeof(pscore), 1, f);
    while (!feof(f)) {
        fprintf(stdout, "%6d%18s%8d%8d%8d\n",
            score.number, score.name, score.mid, score.final, score.quiz);
        fread(&score, sizeof(pscore), 1, f);
    }
    printf("%s\n", " -----");
    fclose(f);
    return 0;
}
```

이진모드로 파일 f에서  
구조체 pscore 자료 읽기

# 파일 접근 처리

- ◆ 파일 위치 (file position)
  - 파일 내부를 byte 단위로 나타내는 값
- ◆ 파일의 순차적 접근
  - 파일 위치는 기본적으로 0부터 시작하여 read 또는 write할 때마다 1 byte씩 뒤로 가는 순차 접근 방식
- ◆ 파일의 임의 접근
  - 아래 함수들 이용하여 임의 접근 가능
  - `int fseek( FILE *fp, long offset, int origin );`
    - ❖ 파일 위치를 origin으로부터 offset만큼 이동
  - `long ftell( FILE *fp );` // 파일 위치를 반환
  - `void rewind( FILE *fp );` // 파일 위치를 0으로 이동
- ◆ 파일 열기 모드 (이진 파일 : b 추가)
  - r (읽기모드) : 쓰기 불가
  - w (쓰기모드) : 읽기 불가
  - a (추가모드) : 파일 끝에만 쓸 수 있고 읽기 불가
  - + (수정 모드) : 원래의 모드에서 읽기 또는 쓰기가 추가됨  
모드 전환 시 `fflush()`, `fseek()`, `rewind()` 같은 위치 함수 호출 필요
  - R+ : 읽기 모드로 열어 쓰기 모드로 전환 가능. 파일이 없으면 오류.
  - W+ : 쓰기 모드로 열어 읽기 모드로 전환 가능. 파일의 기존 내용 삭제됨.
  - A+ : 추가 모드로 파일을 열어 필요하면 읽기 모드로 전환 가능.

SEEK\_SET,  
SEEK\_END,  
SEEK\_CUR