

# C Programming

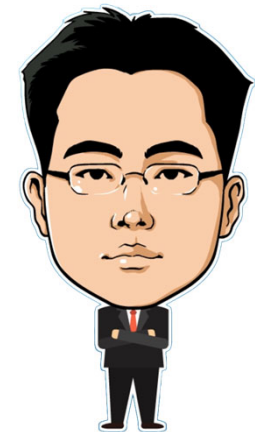
## 파일 처리 (File Processing)



Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



# 목 차



- 파일 입출력의 이해

백문이불여일타(百聞而不如一打)

- 파일 처리: 텍스트 파일

- 파일 처리: 이진 파일

- 다양한 파일 처리 함수



# 파일 입출력의 이해



- 파일 입출력의 이해

백문이불여일타(百聞而不如一打)

- FILE 구조체

- 파일 처리: fopen, fclose

- 파일 처리: 텍스트 파일

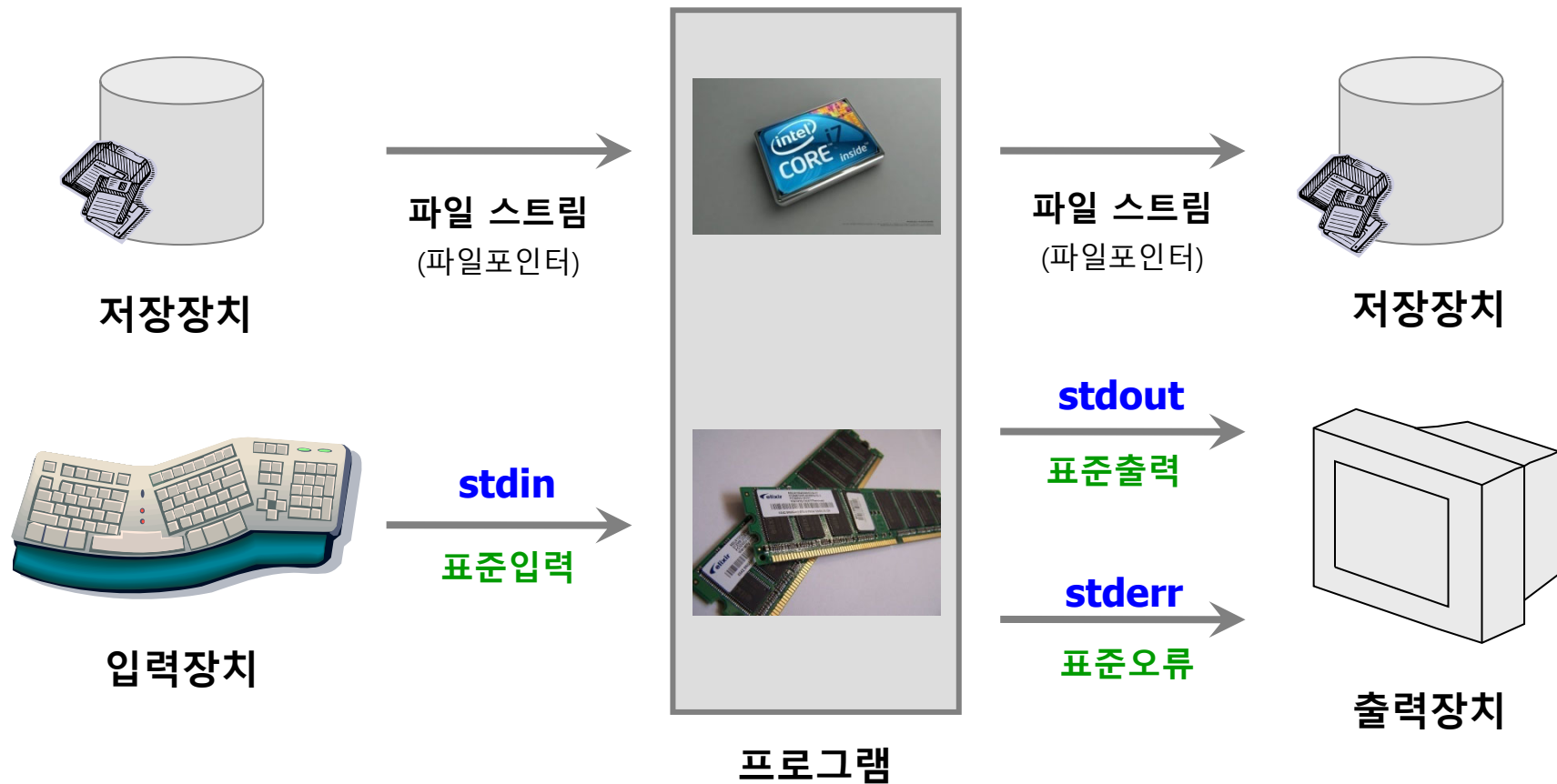
- 파일 처리: 이진 파일

- 다양한 파일 처리 함수



# 파일 입출력의 이해 (1/5)

- **스트림(stream):** 표준 입출력과 파일 입출력



# 파일 입출력의 이해 (2/5)

## ● 스트림(Stream)

### ○ 데이터의 논리적 흐름

- 개발자와 하드웨어 장치 사이에 존재하는 추상적 계층
- 하드웨어 장치 파일과 연결되어 데이터 전송을 중재

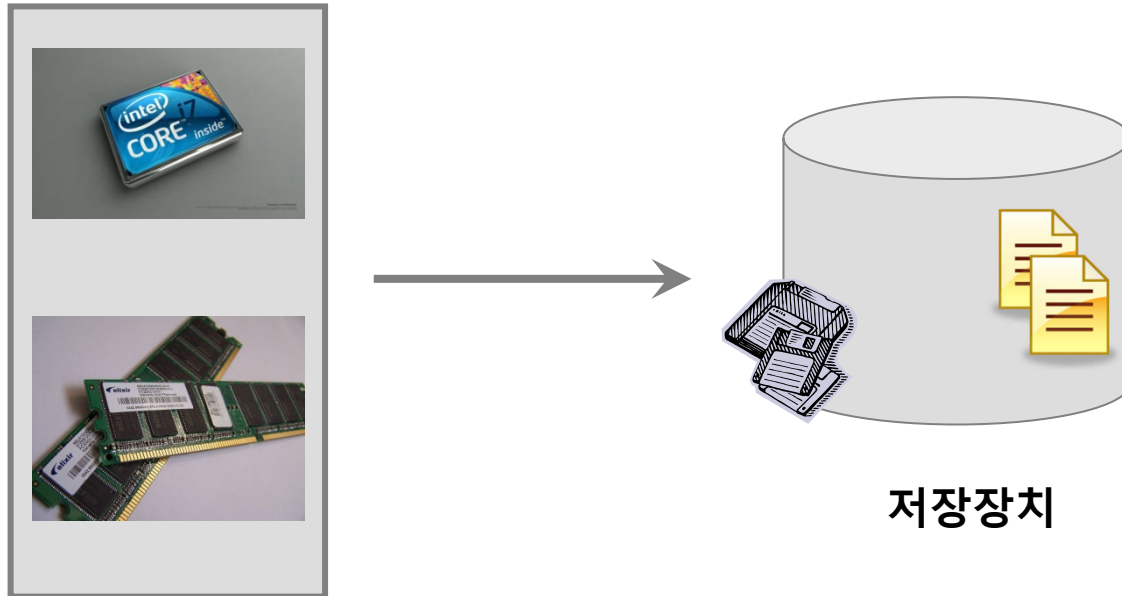
### ○ 표준 입출력 스트림: **<stdio.h>**

입출력 스트림	기능	장치
<b>stdin</b>	표준 입력	키보드
<b>stdout</b>	표준 출력	모니터
<b>stderr</b>	표준 오류	모니터
<b>stdprn</b>	표준 프린터	프린터(LPT1)
<b>stdaux</b>	표준 보조 입출력	직렬 포트(COM1)

# 파일 입출력의 이해 (3/5)

## ● 파일(File)

- 하나의 단위로 취급해야 하는 데이터들의 외부적 컬렉션
- 파일의 종류
  - 텍스트 파일: 모든 데이터가 그래픽 문자로 저장(라인 단위로 구성)
  - 이진 파일: 데이터가 정수 혹은 부동소수점 숫자와 같은 컴퓨터 내부 표현 형식으로 저장



# 파일 입출력의 이해 (4/5)

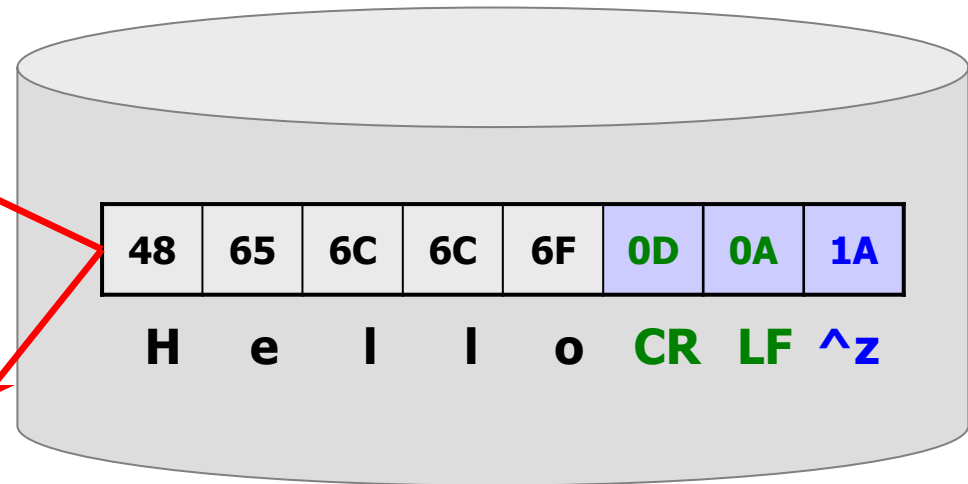
- 텍스트 파일과 이진 파일

텍스트 파일로 읽은 경우

48	65	6C	6C	6F	0A
----	----	----	----	----	----

이진 파일로 읽은 경우

48	65	6C	6C	6F	0D	0A	1A
----	----	----	----	----	----	----	----

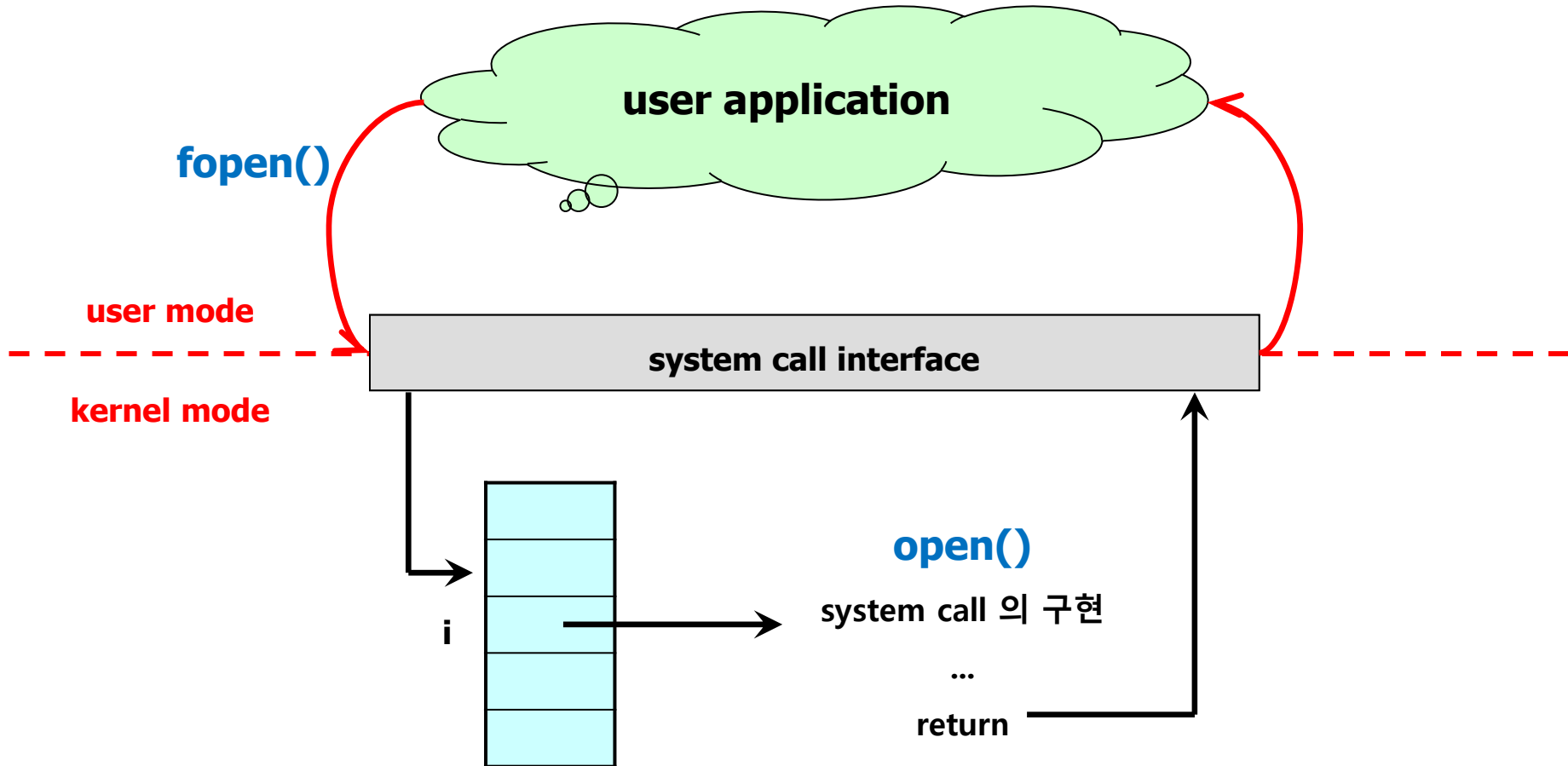


저장장치

“이진 모드로 읽으면 CR과 LF, 그리고 Ctrl-z도 단순한 데이터로 취급하여 그대로 읽혀진다.”

# 파일 입출력의 해 (5/5)

- 시스템 콜(System Call)







# 파일 입출력

FILE 구조체,  
파일 처리: fopen, fclose



# FILE 구조체

- **FILE 구조체:** <stdio.h>

- 파일 또는 텍스트 스트림에 관한 정보를 저장하는 구조체

- 파일 이름, 파일 버퍼의 위치, 파일의 현재 상태 등의 정보를 가지고 있다.

```
// #include <stdio.h>
#ifndef _FILE_DEFINED
struct _iobuf {
    char*    _ptr;           // 파일 포인터: 파일의 현재 위치
    int      _cnt;           // 입력 버퍼에서 사용할 수 있는 문자 개수
    char*    _base;         // 메모리 상에 있는 파일 원형의 주소
    int      _flag;         // 파일 포인터가 파일의 끝에 오면 제5비트가 1이 된다.
    int      _file;         // 파일 식별자
    int      _charbuf;      // 문자열 버퍼
    int      _bufsiz;       // 버퍼의 크기
    char*    _tmpfname;     // 임시 파일 이름의 위치
};
typedef struct _iobuf FILE;
#define _FILE_DEFINED
#endif
```

# 파일 처리: fopen, fclose (1/4)

## ● 파일 개방: fopen / fopen\_s

- 외부 파일과 프로그램 사이에 연결을 만든다.
  - 파일 테이블을 생성하여 파일 처리에 필요한 정보를 저장한다.

```
FILE* fopen(const char* filename, const char* mode); // until C99
FILE* fopen(const char* restrict filename, const char* restrict mode); // since C99
errno_t fopen_s(FILE *restrict* restrict stream,
                const char* restrict filename, const char* restrict mode); // since C11
```

- **filename:** 대상 파일의 경로를 포함한 이름 지정
- **mode:** 파일 개방 모드 설정(r, w, a 등)
- **stream:** 파일 포인터

## ● 파일 닫기: fclose

- 버퍼 공간과 같은 파일 관련 시스템 자원을 반납한다.

```
int fclose(FILE* stream);
```

호출 성공: 0 값을 반환

호출 실패: EOF 값을 반환

# 파일 처리: fopen, fclose (2/4)

## ● 파일 처리: fopen / fclose

```
#include <stdio.h>
#include <stdlib.h>                // exit, EXIT_SUCCESS, EXIT_FAILURE

int main(void)
{
    FILE* fp = NULL;

    // 1) 파일 개방
    // fp = fopen("data.txt", "w");
    fopen_s(&fp, "data.txt", "r");
    if (fp == NULL) {              // 에러 처리!!!
        // fprintf(stderr, "fopen() failed in file %s at line # %d", __FILE__, __LINE__);
        // perror("File opening failed");
        printf("파일 개방 실패!!! \n");
        exit(1);                  // exit(EXIT_FAILURE);;
    }

    // 2) 대상 파일에 대한 읽고 쓰는 작업을 수행: 파일 입출력 함수

    // 3) 파일 닫기
    fclose(fp);

    return 0;                      // return EXIT_SUCCESS;
}
```

# 파일 처리: fopen, fclose (3/4)

- **파일 개방: 텍스트 모드**

- fopen 함수의 파일 개방: **텍스트 모드(Text Mode)**

모드	의 미	비고
r	<b>읽기 전용(Read Only)</b> 지정된 파일이 존재하지 않으면 NULL 값을 반환한다.	쓰기 불가
w	<b>쓰기 전용(Write Only)</b> 파일이 존재할 경우: 기존 파일의 데이터 삭제 후, 빈 파일을 만든다. 파일이 존재하지 않을 경우 : 새로운 파일 생성	읽기 불가
a	<b>추가 전용(Append Only)</b> 파일이 존재할 경우: 기존 파일의 맨 끝에서부터 추가한다. 파일이 존재하지 않을 경우 : 새로운 파일 생성	읽기 불가
r+	지정된 파일이 존재할 경우, 읽기와 쓰기가 가능하도록 파일을 개방한다.	읽기 + 쓰기
w+	지정된 파일에 대하여 읽기와 쓰기가 모드 가능하도록 파일을 개방한다.	읽기 + 쓰기
a+	지정된 파일에 대하여 읽기와 쓰기가 모드 가능하도록 파일을 개방한다. 단, 지정된 파일의 맨 끝에서부터 읽기와 쓰기가 가능하다.	읽기 + 쓰기 이전 부분 쓰기 불가

# 파일 처리: fopen, fclose (4/4)

- **파일 개방: 이진 모드**

- fopen 함수의 파일 개방: 이진 모드(Binary Mode)

모 드	의 미
<b>rb</b> <b>wb</b> <b>ab</b>	이진 모드로 파일을 개방하고, 텍스트 모드에서 r, w, a 와 같은 의미
<b>rb+ / r+b</b>	이진 모드로 파일을 개방하고, 텍스트 모드에서 r+, w+, a+ 와 동일한 의미
<b>wb+ / w+b</b>	
<b>ab+ / a+b</b>	

# 파일 처리: 텍스트 파일



- 파일 입출력의 이해

백문이불여일타(百聞而不如一打)

- 파일 처리: 텍스트 파일

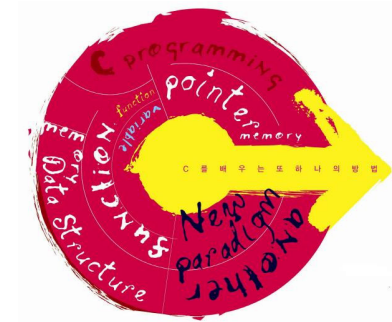
- 문자 파일 입출력: fgetc, fputc

- 문자열 파일 입출력: fgets, fputs

- 서식화 된 파일 입출력: fscanf, fprintf

- 파일 처리: 이진 파일

- 다양한 파일 처리 함수





## 파일 처리: 텍스트 파일

문자 파일 입출력: `fgetc`, `fputc`





# 문자 파일 입출력: fgetc, fputc (1/4)

- 문자 입출력 함수: fgetc, fputc

- fgetc 함수: 파일에서 한 문자(1byte)씩 읽는 함수

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

```
int fgetc (FILE* stream);
```

반환(성공): 읽어 들인 한 문자의 byte를 int 형으로 반환  
(실패 또는 파일의 끝을 만나면): EOF 를 반환

- fputc 함수: 파일에 한 문자(1byte)씩 쓰는 함수

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

```
int fputc (int ch, FILE* stream);
```

반환(성공): 기록한 문자를 int 형으로 반환  
(실패): EOF 를 반환

# 문자 파일 입출력: fgetc, fputc (2/4)

- 문자 입출력 함수: `ungetc`

- 지정된 문자를 입력 스트림에 되돌려 놓는다(`getc` 함수와 반대 동작).
  - 이 스트림을 다음에 읽으면 지금 되돌려진 문자가 반환될 것이다.

```
#include <stdio.h>
int ungetc (int ch, FILE* stream);
```

```
ch = getc(stdin);
if(isdigit(ch))
    ungetc(ch, stdin);
else {
    ... }
```

# 문자 파일 입출력: fgetc, fputc (3/4)

## 예제 10-1: 텍스트 파일 입출력 -- fputc 함수

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp = NULL;

    // fp = fopen("data.txt", "w");
    fopen_s(&fp, "data.txt", "w");
    if (fp == NULL) {
        printf("data.txt 파일 개방 실패!!! \n");
        exit(1);
    }

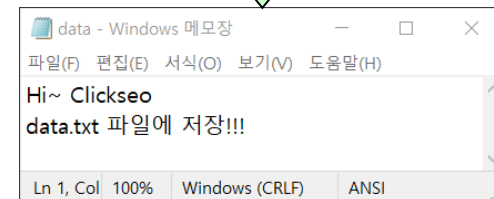
    char ch;
    printf("파일에 저장할 문장 입력(종료: Ctrl + Z)... \n");
    // 사용자로부터 데이터 입력: 표준 입력(stdin)
    while ((ch = getchar()) != EOF)
        fputc(ch, fp); // 파일 버퍼 출력(저장)

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

Microsoft Visual Studio 디버그 콘솔

파일에 저장할 문장 입력(종료: Ctrl + Z)...  
Hi~ Clickseo  
data.txt 파일에 저장!!!  
^Z

파일 저장



# 문자 파일 입출력: fgetc, fputc (4/4)

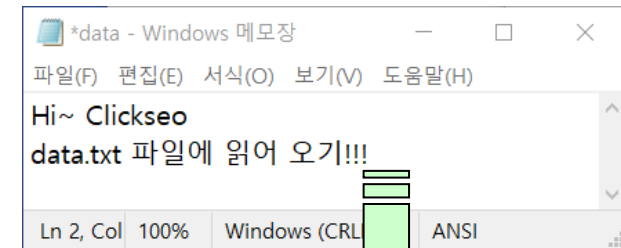
## 예제 10-2: 텍스트 파일 입출력 -- fgetc 함수

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE    *fp;

    // fp = fopen("data.txt", "r");
    fopen_s(&fp, "data.txt", "r");
    if(fp == NULL) {
        printf("data.txt 파일 개방 실패!!! \n");
        exit(1);
    }

    // 파일로 부터 데이터를 읽어와 화면 출력(표준출력, stdout)
    // printf(" ### 파일 내용 ### \n");
    char    ch;
    while( (ch = fgetc(fp)) != EOF )
        putchar(ch);
    putchar('\n');

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



```
Microsoft Visual Studio 디버그 콘솔
### 파일 내용 ###
Hi~ Clickseo
data.txt 파일에 읽어 오기!!!
```



## 파일 처리: 텍스트 파일

문자열 파일 입출력: `fgets`, `fputs`



# 문자열 파일 입출력: fgets, fputs (1/2)

- 문자열 입출력 함수: fgets, fputs

- fgets 함수: 파일에서 문자열 단위로 입력을 수행하는 함수

```
#include <stdio.h>
char* fgets(char* str, int count, FILE* stream); // until C99
char* fgets(char* restrict str, int count, FILE* restrict stream); // since C99
```

반환(성공): str의 주소를 반환  
(실패): NULL 을 반환

- fputs 함수: 파일에서 문자열 단위로 출력을 수행하는 함수

```
#include <stdio.h>
int fputs(const char* str, FILE* stream); // until C99
int fputs(const char* restrict str, FILE* restrict stream); // since C99
```

반환(성공): 음수가 아닌 정수를 반환  
(실패): EOF 를 반환

# 문자열 파일 입출력: fgets, fputs (2/2)

## 예제 10-3: 텍스트 파일 입출력 -- fgets / fputs 함수

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fp = NULL;
    // fp = fopen("data.txt", "w");
    fopen_s(&fp, "data.txt", "w");
    if(fp == NULL) {
        printf("data.txt 파일 개방 실패!!! \n");
        exit(1);
    }

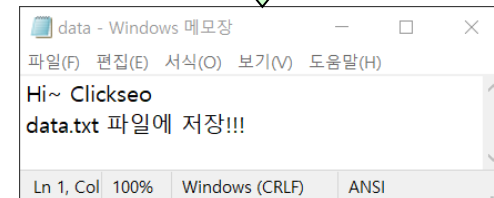
    // 사용자로부터 데이터를 읽어와 파일 출력(저장)
    char str[1024];
    printf("파일에 저장할 문장 입력(종료: Ctrl + Z)... \n");
    while(fgets(str, sizeof(str), stdin))
        fputs(str, fp);

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

Microsoft Visual Studio 디버그 콘솔

파일에 저장할 문장 입력(종료: Ctrl + Z)...  
Hi~ Clickseo  
data.txt 파일에 저장!!!  
^Z

파일 저장





## 파일 처리: 텍스트 파일

서식화 된 파일 입출력: `fscanf`, `fprintf`





# 서식화 된 파일 입출력: fscanf, fprintf (1/2)

- 서식화 된 파일 입출력 함수: fscanf, fprintf 계열 함수

- fscanf / fscanf\_s 함수: 파일을 대상으로 서식화 된 입력 기능을 지원하는 함수

```
#include <stdio.h>
int fscanf(FILE* stream, const char* format, ... );           // until C99
int fscanf(FILE* restrict stream, const char* restrict format, ... ); // since C99
int fscanf_s(FILE* restrict stream, const char* restrict format, ... ); // since C11
```

반환(성공): 정수를 반환  
(실패): EOF 를 반환

- fprintf 함수: 파일을 대상으로 서식화 된 출력 기능을 지원하는 함수

```
#include <stdio.h>
int fprintf(FILE* stream, const char* format, ... );           // until C99
int fprintf(FILE* restrict stream, const char* restrict format, ... ); // since C99
int fprintf_s(FILE* restrict stream, const char* restrict format, ... ); // since C11
```

반환(성공): 정수를 반환  
(실패): EOF 를 반환

# 서식화 된 파일 입출력: fscanf, fprintf (2/2)

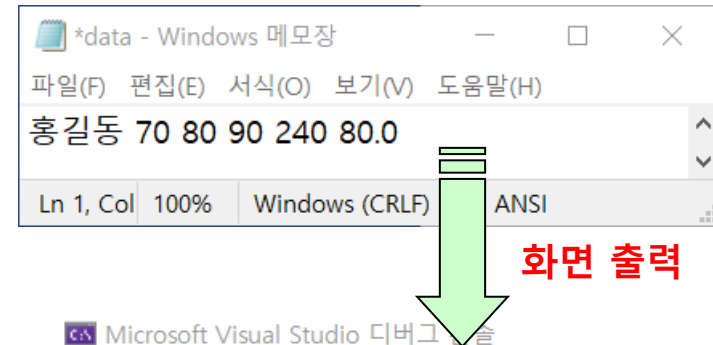
## 예제 10-4: 텍스트 파일 입출력 -- fscanf / fprintf 함수

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char    name[12];
    int     kor, eng, math, tot;
    double  ave;

    FILE    *fp = NULL;
    // fp = fopen("data.txt", "r");
    fopen_s(&fp, "data.txt", "r");
    if(fp == NULL) {
        printf("data.txt 파일 개방 실패!!! \n");
        exit(1);
    }

    // 파일로 부터 데이터를 읽어와 화면 출력(표준출력, stdout)
    // while( fscanf(fp, "%s %d %d %d", name, &kor, &eng, &math) == 4 ) {
    while( fscanf_s(fp, "%s %d %d %d", name, (int)sizeof(name), &kor, &eng, &math) == 4) {
        tot = kor + eng + math;
        ave = tot / 3.0;
        printf("%10s %3d %3d %3d %5d %8.2f \n", name, kor, eng, math, tot, ave);
    }

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



# 파일 처리: 이진 파일



- 파일 입출력의 이해

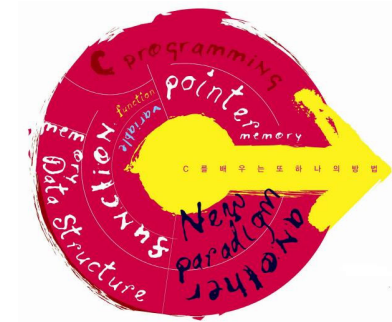
백문이불여일타(百聞而不如一打)

- 파일 처리: 텍스트 파일

- 파일 처리: 이진 파일

- 이진 파일 입출력: fread, fwrite

- 다양한 파일 처리 함수



# 이진 파일 입출력: fread, fwrite (1/3)

## ● 이진 파일 입출력 함수 : fread, fwrite

- fread 함수: 이진 모드로 데이터를 읽는 함수

```
size_t fread(void* buffer, size_t size, size_t count, FILE *stream); // until C99
size_t fread(void* restrict buffer, size_t size, size_t count, // since C99
FILE* restrict stream);
```

호출 성공: 블록의 개수를 반환  
호출 실패: 0 을 반환

- fwrite 함수: 이진 모드로 데이터를 쓰는 함수

```
size_t fwrite(const void* buffer, size_t size, size_t count, // until C99
FILE* stream);
size_t fwrite(const void* restrict buffer, size_t size, size_t count, // since C99
FILE* restrict stream);
```

- **buffer** : 파일로부터 읽어 들인 데이터를 기억시킬 버퍼를 가리키는 포인터
- **size** : 한 번에 읽어 들일 수 있는 데이터의 바이트 수
- **count** : size 만큼 읽어 들이기 위해 지정하는 반복 횟수
- **stream** : 대상이 되는 파일 포인터

호출 성공: 블록의 개수를 반환(n)  
호출 실패: count 이외의 값을 반환

# 이진 파일 입출력: fread, fwrite (2/3)

## 예제 10-5: 이진 파일 입출력 -- fread / fwrite 함수

(1/2)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>    // _getch

typedef struct _score {
    char    name[12];
    int     kor, eng, math, tot;
    float   ave;
} SCORE;
```

```
char* file = "data.bin";
```

```
int main(void)
```

```
{
```

```
    FILE*   fp = NULL;
```

```
    // 파일 개방: 새로운 이진파일 생성(읽기+쓰기 모드)
```

```
    // fp = fopen(file, "wb+");
```

```
    fopen_s(&fp, file, "wb+");
```

```
    if (fp == NULL) {
```

```
        printf("파일 개방 실패!!! \n");
```

```
        exit(1);
```

```
    }
```

Microsoft Visual Studio 디버그 콘솔

이름: 홍길동  
국어: 70  
영어: 80  
수학: 91

> 저장된 파일 정보 출력을 원하면 아무 키나 누르시오...

홍길동 70 80 91 241 80.33

# 이진 파일 입출력: fread, fwrite (3/3)

예제 10-5: 이진 파일 입출력 -- fread / fwrite 함수

(2/2)

```
SCORE temp;
printf("이름: ");      gets_s(temp.name, sizeof(temp.name));
printf("국어: ");     scanf_s("%d", &temp.kor);
printf("영어: ");     scanf_s("%d", &temp.eng);
printf("수학: ");     scanf_s("%d%c", &temp.math);
temp.tot = temp.kor + temp.eng + temp.math;
temp.ave = (float)temp.tot / 3;

// 사용자로부터 입력 받은 데이터를 이진 파일에 저장
fwrite(&temp, sizeof(SCORE), 1, fp);

printf("\n> 저장된 파일 정보 출력을 원하면 아무 키나 누르시오... ");
_getch();      // getch();

// fseek(fp, 0, SEEK_SET);
rewind(fp);

// 파일에서 데이터 읽기: 이진 파일
int count = (int)fread(&temp, sizeof(SCORE), 1, fp);
if(count == 0) printf("파일에서 정보 읽기 실패!!! \n");
else printf("\n\n%s %3d %3d %3d %5d %8.2f\n",
           temp.name, temp.kor, temp.eng, temp.math, temp.tot, temp.ave);
fclose(fp);
return 0;
}
```

# 다양한 파일 처리 함수



- 파일 입출력
- 파일 처리: 텍스트 파일
- 파일 처리: 이진 파일
- 다양한 파일 처리 함수

백문이불여일타(百聞而不如一打)

- 랜덤 액세스 함수: fseek, ftell
- 오류 처리 관련 함수
- 기타 함수



# 다양한 파일 처리 함수 (1/5)

- 랜덤 액세스 함수: `fseek`, `ftell`

- `fseek` 함수: 파일 포인터 `stream`을 임의의 위치로 이동하는 함수
  - 원하는 자료에 대한 직접 접근(direct access)이 가능

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

```
int      fseek(FILE* stream, long offset, int origin);
```

- **stream** : 대상이 되는 파일 포인터
- **offset** : whence 위치부터 새로운 위치까지 상대적으로 떨어진 거리(**bytes** 단위)
- **origin** : 파일 포인터 이동을 위한 기준점

기준점	값	의 미
<code>SEEK_SET</code>	0	파일의 시작위치를 기준으로 파일 포인터를 이동
<code>SEEK_CUR</code>	1	현재의 파일 포인터 위치를 기준으로 다음 위치로 이동
<code>SEEK_END</code>	2	파일의 마지막 위치를 기준으로 파일 포인터를 이동



# 다양한 파일 처리 함수 (2/5)

- 랜덤 액세스 함수: `ftell`

- `ftell` 함수: 파일 포인터의 위치를 얻어온다.

- 파일의 시작부터 현재 stream 위치 까지의 거리 (bytes 단위)

```
long ftell(FILE* stream);
```

반환(성공): 파일의 시작부터 현재 파일 포인터까지의 거리  
(실패): -1 을 반환

```
int count = fread(&buffer, sizeof(double), 1, fp);
pos = ftell(fp);
if (pos == -1L) {
    perror("ftell()");
    fprintf(stderr, "ftell() failed in file %s at line # %d\n",
            __FILE__, __LINE__ - 4);
    exit(EXIT_FAILURE);
}
printf("%ld\n", pos);
```

# 다양한 파일 처리 함수 (3/5)

- 오류 처리 관련 함수: feof, ferror, clearerr

- feof 함수: 파일의 끝에 도달했는지 여부를 알려준다.

```
int feof(FILE* stream);
```

반환(성공): 파일의 끝인 경우 0 이 아닌 값을 반환  
(실패): 0 값을 반환

- ferror 함수: 파일 처리 도중 입출력 오류 발생시 오류를 알려준다.

```
int ferror(FILE* stream);
```

반환(성공): 오류가 없는 경우 0 값을 반환  
(실패): 오류가 존재하는 경우 0 이 아닌 값을 반환

- 파일 처리 도중에 발생할 수 있는 입출력 오류
  - 읽기 오류(read error), 쓰기 오류(write error), 파일 끝(end of file)

- clearerr 함수: error 상태를 clear 하는 함수

- FILE 구조체의 멤버 중 flag의 EOF Indicator 비트를 0으로 설정하고, ferror 함수가 설정해 놓은 Error Indicator 비트도 0으로 만들어 오류 상태를 되돌려 놓는 역할을 한다.

```
void clearerr(FILE* stream);
```

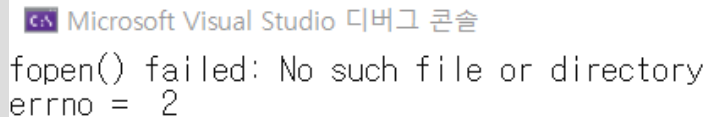
# 다양한 파일 처리 함수 (4/5)

## ● 오류 처리 관련 함수

- perror 함수: 입출력 오류가 발생할 경우에 오류 메시지를 콘솔로 출력
  - 파일이 없거나 이미 존재하는 경우 등 다수의 상황에 따른 오류 코드를 **error.h** 에 매크로 상수로 정의해 두는데, 오류가 발생할 경우 **errno** 에 자동으로 저장된다.

```
void perror(const char *errmsg);
```

```
#include <stdio.h>
#include <errno.h>           // errno
int main(void)
{
    FILE* fp = NULL;
    // fp = fopen("non_existent", "r");
    fopen_s(&fp, "data.txt", "r");
    if(fp == NULL) {
        perror("fopen() failed");
        printf("errno = % d\n", errno);
    }
    else {
        fclose(fp);
    }
    return 0;
}
```



```
Microsoft Visual Studio 디버그 콘솔
fopen() failed: No such file or directory
errno = 2
```

# 다양한 파일 처리 함수 (5/5)

- 기타 함수: fflush

- fflush 함수: 대상이 되는 파일 버퍼를 비우는 함수

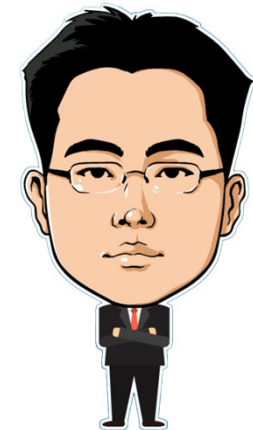
```
int fflush(FILE *stream);
```

반환(성공): 0 값을 반환  
(실패): EOF 값을 반환

- stream이 stdout인 경우는 입출력 버퍼를 비우고, stream이 디스크 파일 스트림인 경우는 파일 버퍼를 비움으로써 버퍼의 내용을 최종적으로 디스크에 기록한다.
- 단, Visual Studio 2015 이상에서는 fflush 함수가 적용되지 않는다.

# 참고문헌

- [1] 서현우, "혼자 공부하는 C 언어 : 1:1 과외 하듯 배우는 프로그래밍 자습서", 한빛미디어, 2023.
- [2] Paul Deitel, Harvey Deitel, "C How to Program", Global Edition, 8/E, Pearson, 2016.
- [3] Kamran Amini, 박지윤 번역, "전문가를 위한 C : 동시성, OOP부터 최신 C, 고급 기능까지!", 한빛미디어, 2022.
- [4] 서두옥, "(열혈강의) 또 하나의 C : 프로그래밍은 셀프입니다", 프리렉, 2012.
- [5] Behrouz A. Forouzan, Richard F. Gilberg, 김진 외 7인 공역, "구조적 프로그래밍 기법을 위한 C", 도서출판 인터비전, 2004.
- [6] Brian W. Kernighan, Dennis M. Ritchie, 김석환 외 2인 공역, "The C Programming Language", 2/E, 대영사, 2004.
- [7] "C reference", cppreference.com, 2023 of viewing the site, <https://en.cppreference.com/w/c>.



이 강의자료는 저작권법에 따라 보호받는 저작물이므로 무단 전제와 무단 복제를 금지하며, 내용의 전부 또는 일부를 이용하려면 반드시 저작권자의 서면 동의를 받아야 합니다.

Copyright © Clickseo.com. All rights reserved.

