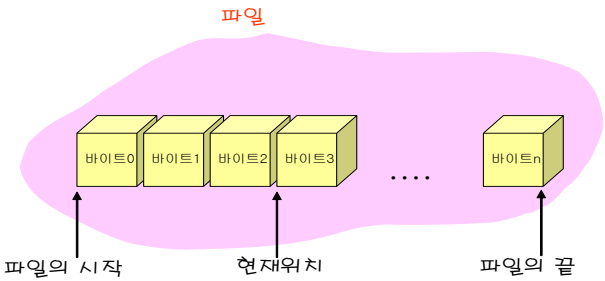
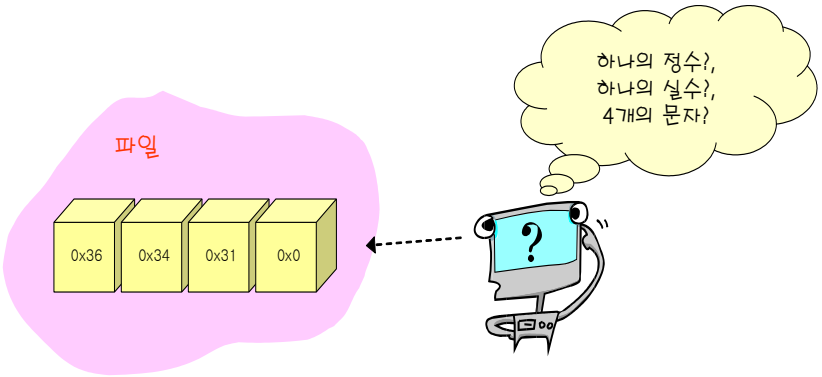
C언어를 이용한 화일처리

C에서의 화일의 개념

- C에서의 화일은 일련의 연속된 바이트
- 모든 데이터들은 결국은 바이트(1문자)로 바뀌어서 화일에 저장

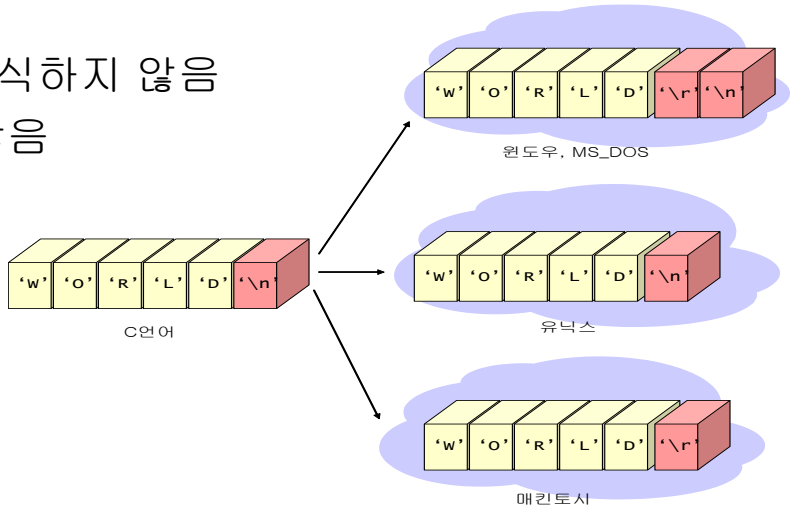


- 화일에 4개의 바이트가 들어 있을 때
  - 정수 데이터로도 해석할 수 있고
  - float형 실수 데이터
  - 문자형으로 해석 가능
- 이들 바이트들을 어떻게 해석하느냐는 전적으로 프로그램의 지시에 따라 달라짐



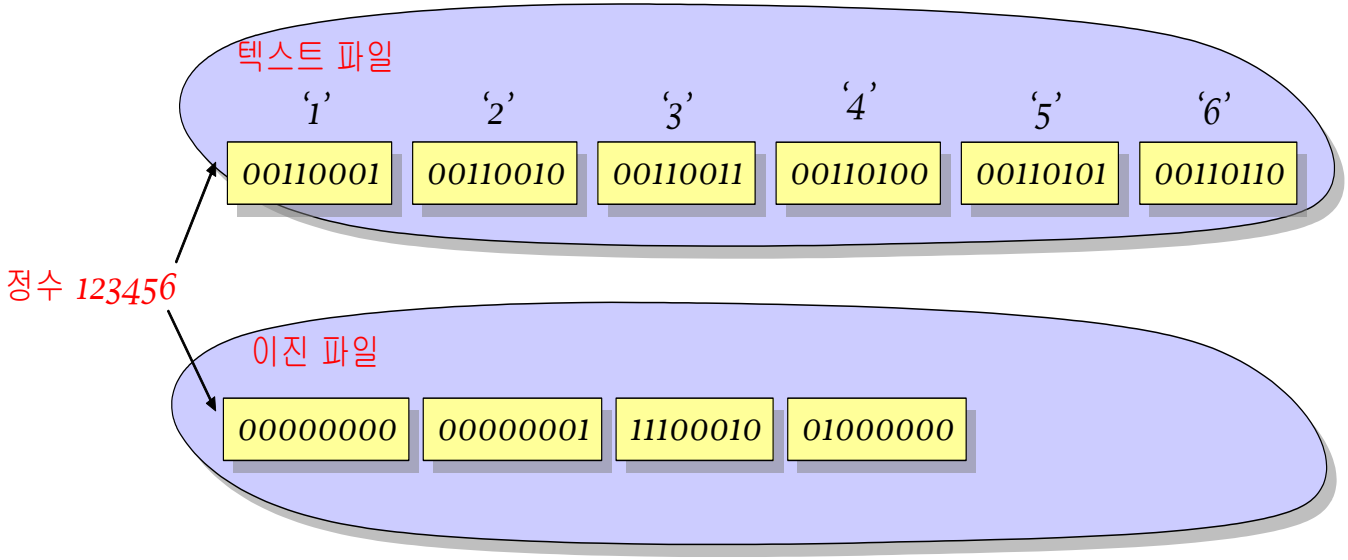
화일의 종류

- 크게 두가지로 구분
  - 택스트화일(text file) vs. 이진화일(binary file)
- 텍스트 화일
  - 사람이 읽을 수 있는 텍스트가 들어 있는 화일
    - C 프로그램 소스 화일이나 메모장 화일
- 이진 화일
  - 사람이 읽을 수는 없으나 컴퓨터는 읽을 수 있는 화일
  - 이진 데이터가 직접 저장되어 있는 화일
  - 이진 화일은 특정 프로그램에 의해서만 판독이 가능
    - C 프로그램 실행 화일, 사운드 화일, 이미지 화일
- 차이점
  - 택스트화일은 제어코드를 데이터로 인식하지 않음
  - 이진화일은 모든 데이터를 구분하지 않음



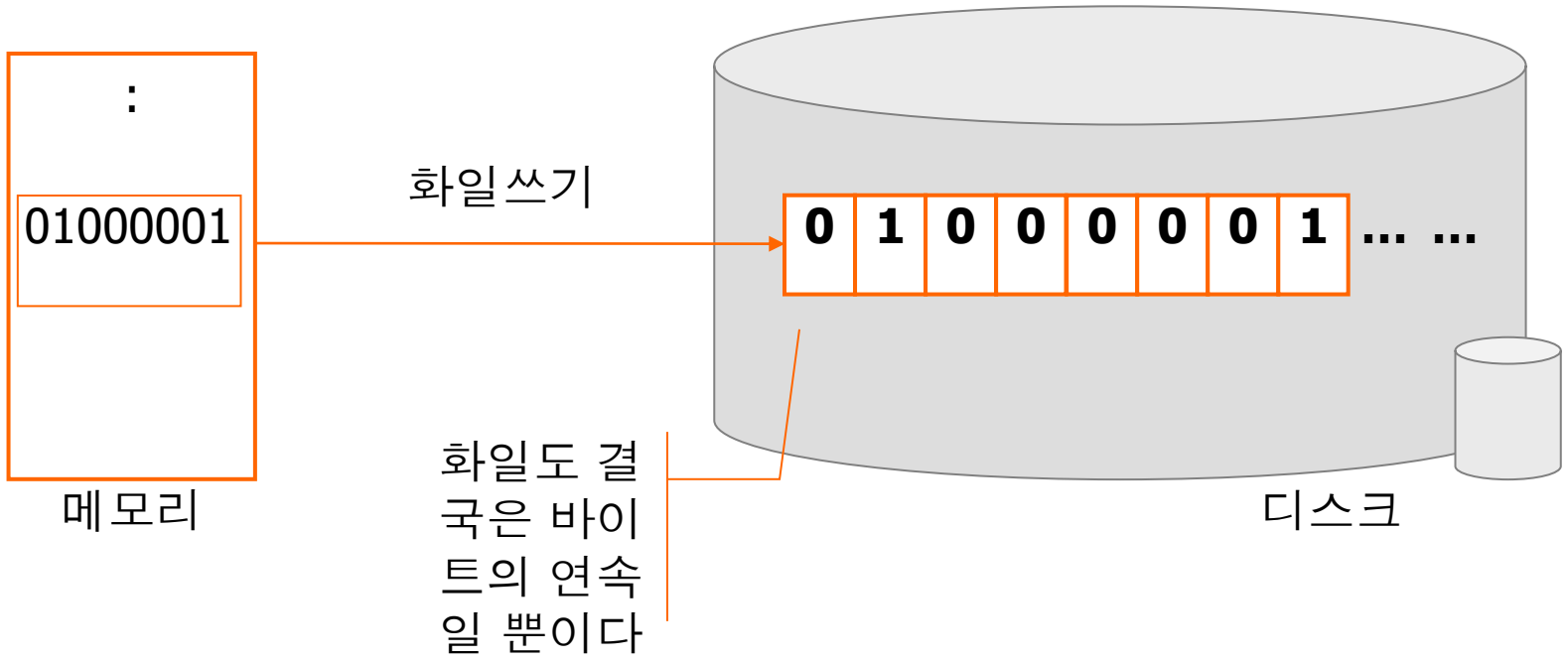
데이터 저장 시 주의점

- 정수 12345를 저장하는 방법
  - fprintf(fp, "%d ", 12345);
    - *텍스트화일*로 저장됨
    - 모든 데이터가 아스키 코드로 변환되어서 저장됨
  - fwrite(12345,fp);
    - *이진화일*로 저장됨
    - 컴퓨터에서 데이터를 표현하는 방식 그대로 저장



화일저장구조

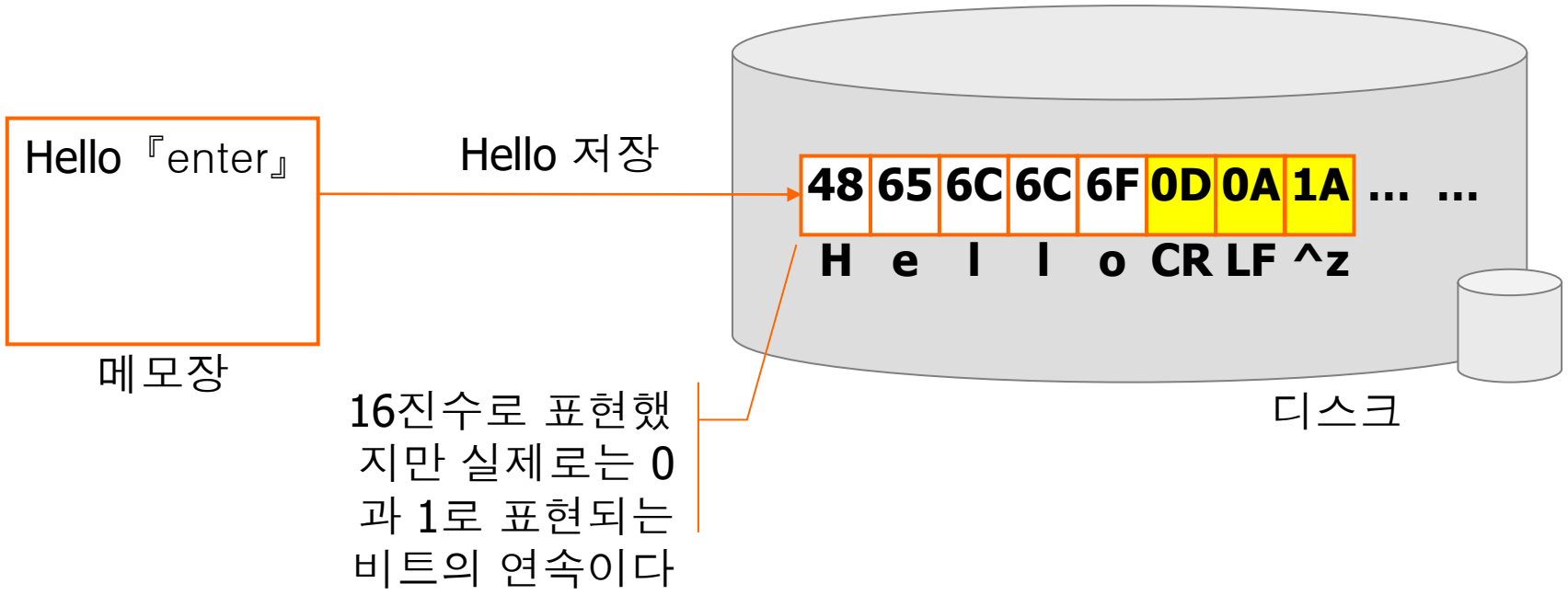
❖ 화일은 디스크에 어떤 모습으로 존재하는가?



- 메모리와 화일의 관계

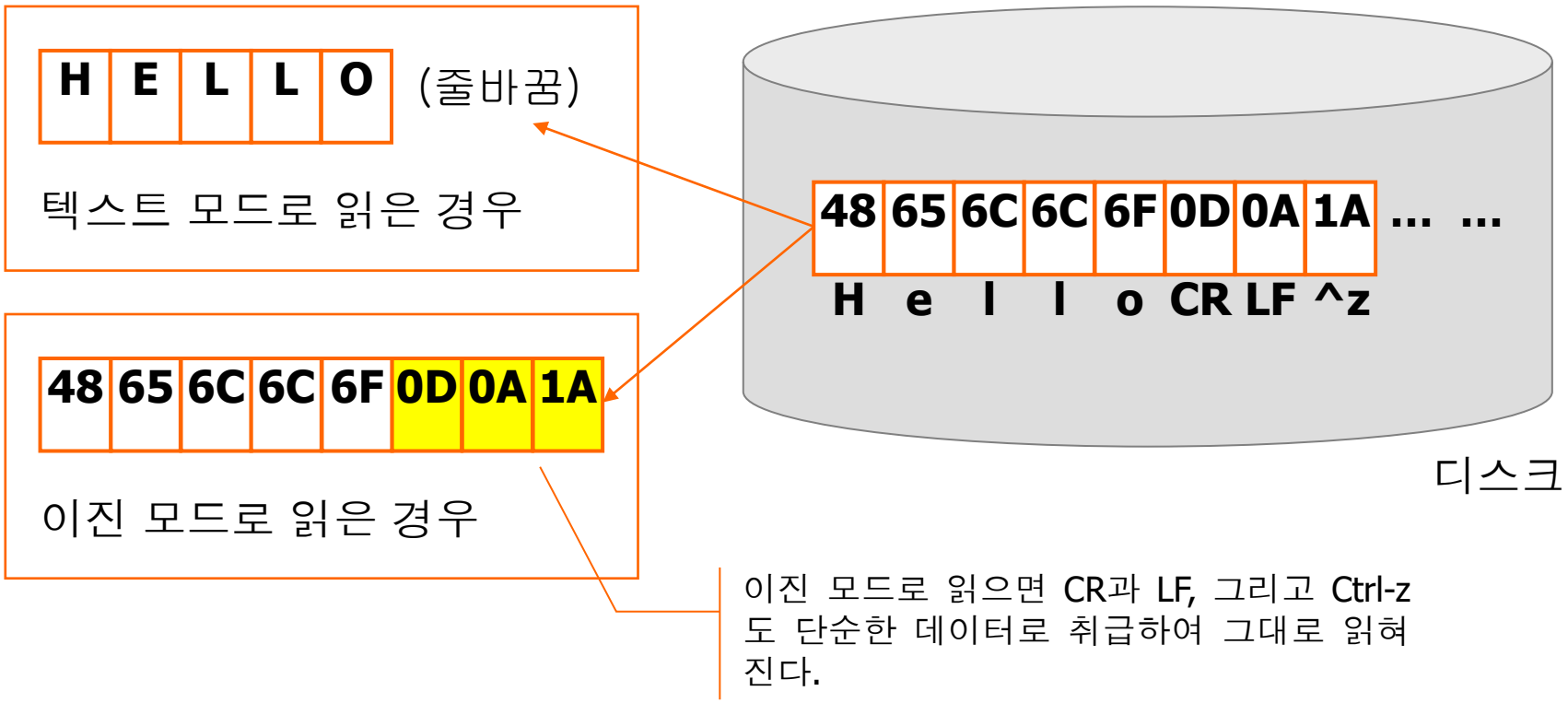
화일저장구조

❖ 텍스트 화일과 이진화일

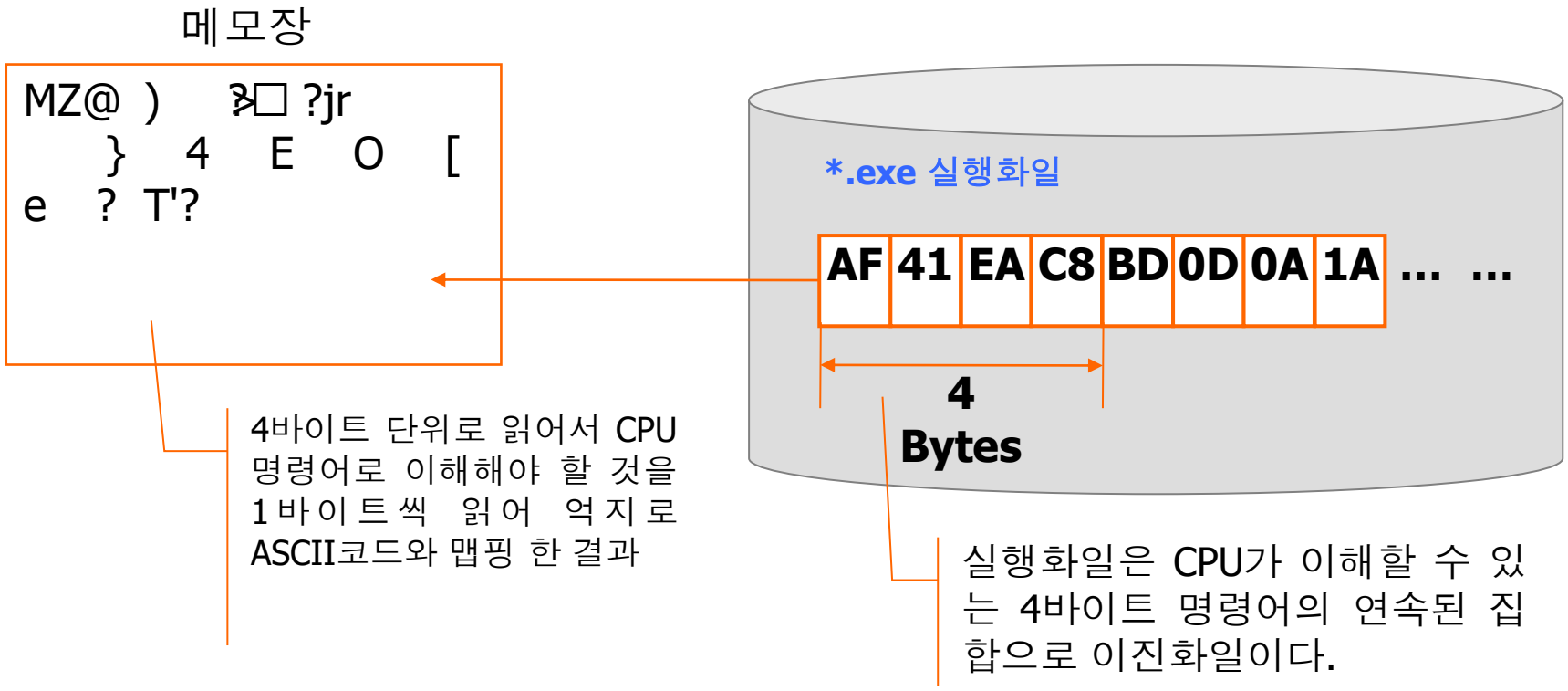


- 텍스트 화일의 저장

텍스트 모드와 이진모드의 차이 1



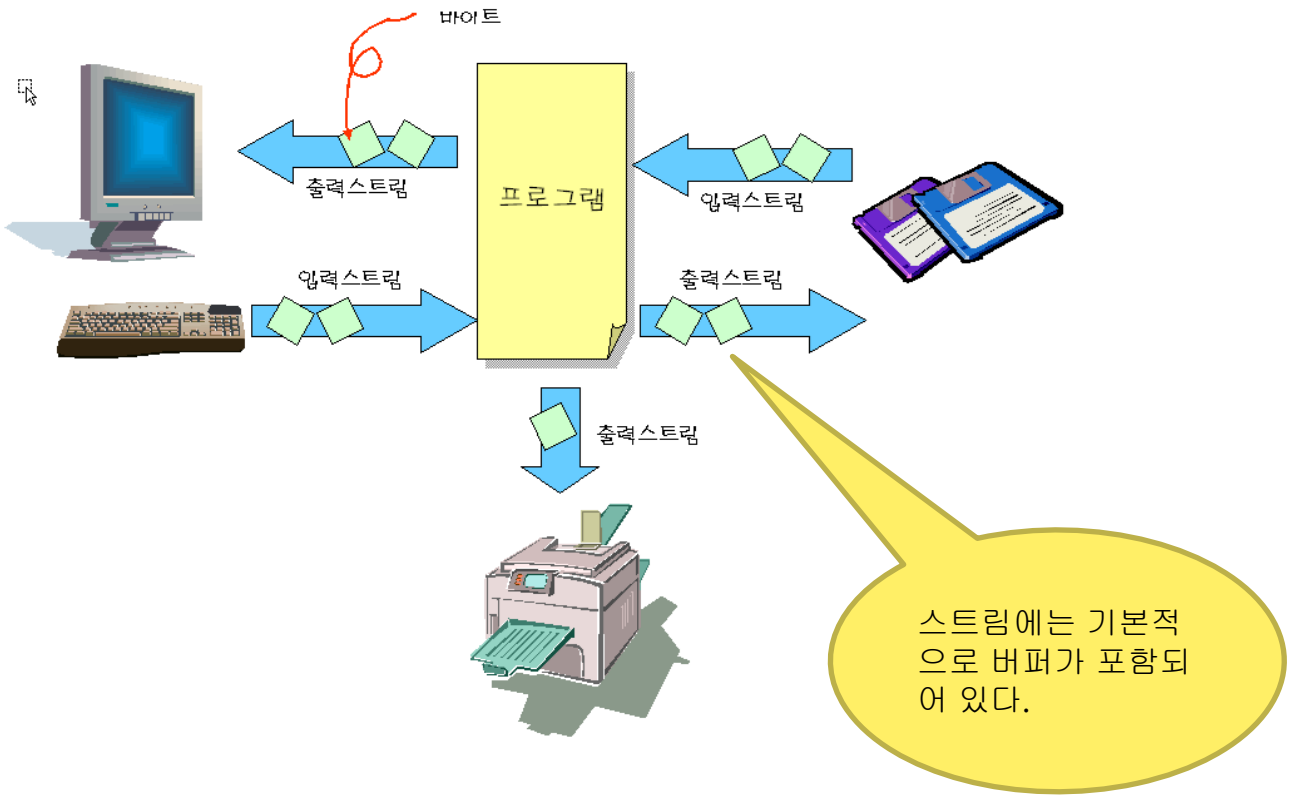
텍스트 모드와 이진모드의 차이 2



화일 입출력의 개념

스트림의 이해

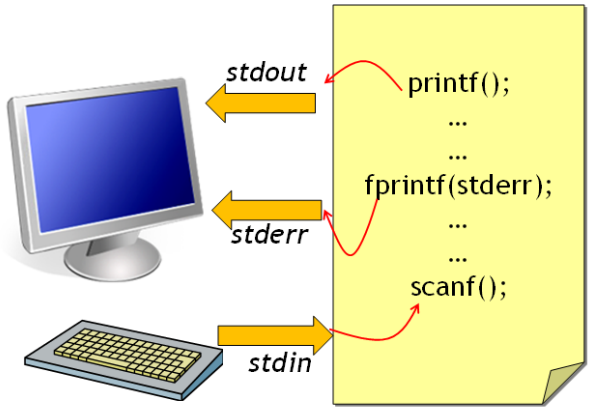
- 스트림(stream)
  - 입력과 출력을 바이트(byte)들의 흐름으로 생각하는 것



표준 입출력 스트림

- 컴퓨터는 모든 외부장치를 화일(디바이스 화일)로 간주한다
  - 따라서 표준입출력장치는 다음과 같은 스트림 이름을 갖는다

이름	*스트림*	*연결 장치*
stdin	표준 입력 스트림	키보드
stdout	표준 출력 스트림	모니터의 화면
stderr	표준 오류 스트림	모니터의 화면



입출력 함수의 분류

- 사용하는 스트림에 따른 분류
  - 표준 입출력 스트림
  - 일반 입출력 스트림

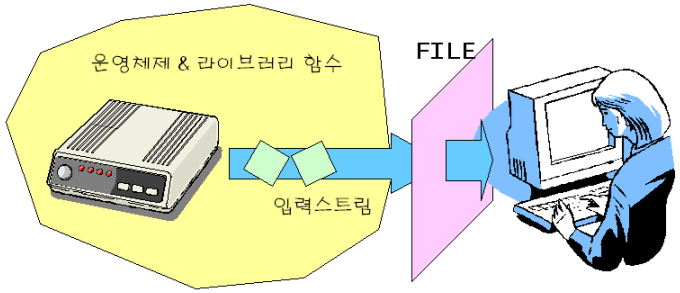
스트림 형식	표준 스트림	일반 스트림	설명
형식이 없는 입출력 (문자 형태)	getchar()	fgetc(FILE *f,...)	문자 입력 함수
	putchar()	fputc(FILE *f,...)	문자 출력 함수
	gets()	fgets(FILE *f,...)	문자열 입력 함수
	puts()	fputs(FILE *f,...)	문자열 출력 함수
형식이 있는 입출력 (정수, 실수,..)	printf()	fprintf(FILE *f,...)	형식화된 출력 함수
	scanf()	fscanf(FILE *f,...)	형식화된 입력 함수

표준입출력장치, 즉, 키보드' 모니터에 의한 입출력

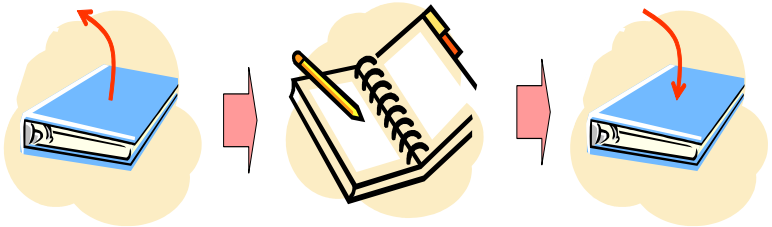
일반적인 화일입출력

화일의 입출력

- 스트림은 구체적으로 **FILE** 구조체를 통하여 구현
- **FILE**은 **stdio.h**에 정의되어 있다.
- 디스크 화일은 **FILE** 구조체를 이용하여 접근
- **FILE** 구조체를 가리키는 포인터를 화일 포인터(**file pointer**)



- 화일을 다룰 때는 반드시 다음과 같은 순서를 지켜야 한다.



화일 열기/닫기

- 화일 열기
  - 화일에서 데이터를 읽거나 쓸 수 있도록 모든 준비를 마치는 것

```
FILE *fopen(char *name, char *mode)
```

  - 첫 번째 매개 변수인 **name**은 화일의 이름
  - 두 번째 매개 변수인 **mode**는 화일을 여는 모드를 의미
    - 읽기, 쓰기, 추가

- 화일 닫기

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

화일 열기 모드

모드		의 미	비 고
텍스트모드	r	읽기전용(read only)으로 열고 화일이 존재하지 않으면 NULL을 반환한다.	write 불가
	w	쓰기전용(write only)으로 열고 화일이 존재하지 않으면 새 화일을 생성하고 기존화일이 존재하면 그 내용은 무시하고 처음부터 새로 쓴다.	read 불가
	a	추가모드(append only)로 열고 화일이 존재하지 않으면 새 화일을 생성하고 존재하면 기존 화일의 끝에 추가만 가능	read 불가
	r+	기존 화일에 대한 읽기와 쓰기가 모두 가능하도록 화일을 열고 화일이 존재하지 않으면 NULL을 반환	read + write
	w+	무조건 새로운 화일을 생성하여 읽기와 쓰기 모두 가능하도록 화일을 연다. 화일이 존재하지 않으면 새로 생성한다.	read + write
	a+	기존 화일의 끝에서부터 읽기와 쓰기가 가능하도록 화일을 열고 화일이 존재하지 않으면 새로 생성한다.	이전부분은 write 불가
이진모드	rb, wb, ab	이진 모드(binary mode)로 화일을 개방하고 텍스트 모드에서의 r, w, a와 같은 의미를 가짐	
	r+b, rb+	이진 모드로 화일을 개방하고 텍스트 모드에서의 r+, w+, a+와 동일한 의미를 가짐	
	w+b, wb+		
	a+b, ab+		

file_open.c

```c
// 화일 열기
#include <stdio.h>

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

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

	if( fp == NULL )
		printf("화일 열기 실패\n");
	else
		printf("화일 열기 성공\n");

	fclose(fp);

	return 0;
}
```

```
화일 열기 성공
```

화일 입출력 함수

- 화일 입출력 라이브러리 함수

종류	설명	입력 함수	출력 함수
문자 단위	문자 단위로 입출력	int fgetc(FILE *fp)	int fputc(int c, FILE *fp)
문자열 단위	문자열 단위로 입출력	char *fgets(FILE *fp)	int fputs(const char *s, FILE *fp)
서식화된 입출력	형식 지정 입출력	int fscanf(FILE *fp, ...)	int fprintf(FILE *fp,...)
이진 데이터	이진 데이터 입출력	fread()	fwrite()

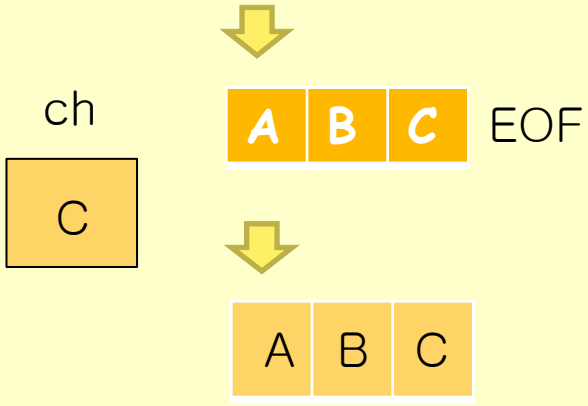
```
int  fgetc(FILE *fp);
int  fputc(int ch, FILE *fp);
                - ch : 화일에 기록할 문자 상수
                - fp : 대상이 되는 화일 포인터
```

```
fgetc(stdin);          /* getchar() 와 동일
fputc('H', stdout);    /* putchar()와 동일*/
```

```
char *fgets(char *str, int n, FILE *fp);
                - str : 화일에서 읽어 들인 문자열을 저장할 공간에 대한 포인터
                - n : 읽어 들일 문자열의 최대 길이
                - fp : 대상이 되는 화일 포인터
int  fputs(char *str, FILE *fp);
                - ch : 화일에 기록할 문자열
                - fp : 대상이 되는 화일 포인터
```

화일 복사 예제1

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
        FILE *fp1, *fp2;
        int ch;

        // 첫번째 화일을 읽기 모드로 연다.
        if( (fp1 = fopen("sample.txt", "r")) == NULL )
        {
                fprintf(stderr,"원본 화일을 열 수 없습니다.\n");
                return 1;
        }

        // 두번째 화일을 쓰기 모드로 연다.
        if( (fp2 = fopen("copied.txt", "w")) == NULL )
        {
                fprintf(stderr,"복사 화일을 열 수 없습니다.\n");
                return 2;
        }

        // 첫번째 화일을 두번째 화일로 복사한다.
        while( (ch=fgetc(fp1)) != EOF )
        {
                fputc(ch, fp2);
        }

        fclose(fp1);
        fclose(fp2);
        return 0;
}
```



화일 복사 예제2

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
        FILE *fp1, *fp2;
        char buffer[100];

        // 첫번째 화일을 읽기 모드로 연다.
        if( (fp1 = fopen("proverb.txt", "r")) == NULL )
        {
                fprintf(stderr,"원본 화일 %s을 열 수 없습니다.\n", "proverb.txt");
                return 1;
        }
        // 두번째 화일을 쓰기 모드로 연다.
        if( (fp2 = fopen(" proverb2.txt", "w")) == NULL )
        {
                fprintf(stderr,"복사 화일 %s을 열 수 없습니다.\n", "copied.txt");
                return 2;
        }

        // 첫번째 화일을 두번째 화일로 복사한다.
        while( fgets(buffer, 100, fp1) != NULL )
   {
                fputs(buffer, fp2);
        }

        fclose(fp1);
        fclose(fp2);
        return 0;
}
```

서식화된 화일 입출력 함수 : fscanf() 와 fprintf()

int *fscanf(FILE *fp, char *format, 가변길이인수리스트);

- fp : 대상이 되는 화일 포인터
- format : 서식 문자열
- 가변길이인수리스트 : 화일로부터 읽은 자료를 보관할 변수 목록

int fprintf(FILE *fp, char *format, 가변길이인수리스트);

- fp : 대상이 되는 화일 포인터
- format : 서식 문자열
- 가변길이인수리스트 : 화일로 저장할 내용을 담고 있는 변수 목록

- **택스트모드로 저장한다**
  - **숫자 12345를 저장하면 문자열 '12345'로 변환되어 저장**
- **기본적인 사용법은 scanf()나 printf() 함수와 동일하다.**
  - **단, 화일 포인터를 이용하여 지정된 화일에 저장**

서식화된 입출력

출력 예제

```
int i = 23;
float f = 1.2345;
FILE *fp;

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

if( fp != NULL )
                fprintf(fp, "%10d %16.3f", i, f);

fclose(fp);
```

입력 예제

```
int i;
float f;
FILE *fp;

fp = fopen("sample.txt", "r");

if( fp != NULL )
                fscanf(fp, "%d %f", &i, &f);

fclose(fp):
```

Score_out.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
        FILE *fp;
        int number, count = 0;
        char name[20];
        float score, total = 0.0;
        // 성적 화일을 쓰기 모드로 연다.
        if( (fp = fopen("score.txt", "w")) == NULL )
        {
                fprintf(stderr,"성적 화일을 열 수 없습니다.\n");
                exit(1);
        }
        // 사용자로부터 학번, 이름, 성적을 입력받아서 화일에 저장한다.
        while( 1 )
        {
                printf("학번, 이름, 성적을 입력하시요: (음수이면 종료)");
                scanf("%d", &number);
                if( number < 0 ) break
                scanf("%s %f", name, &score);
                fprintf(fp, "%d %s %f\n", number, name, score);
        }
        fclose(fp);
        return 0;
}
```

```
학번, 이름, 성적을 입력하시요: (음수이면 종료) 1 KIM 90.2
학번, 이름, 성적을 입력하시요: (음수이면 종료) 2 PARK 30.5
학번, 이름, 성적을 입력하시요: (음수이면 종료) 3 MIN 56.8
학번, 이름, 성적을 입력하시요: (음수이면 종료)-1
```

Score_in.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
        FILE *fp;
        int number, count = 0;
        char name[20];
        float score, total = 0.0;
        // 성적 화일을 읽기 모드로 연다.
        if( (fp = fopen((“score.txt”, "r")) == NULL )
        {
                fprintf(stderr,"화일을 열 수 없습니다.\n");
                exit(1);
        }
        // 화일에서 성적을 읽어서 평균을 구한다.
        while( !feof( fp ) )
        {
                fscanf(fp, "%d %s %f", &number, name, &score);
                total += score;
                count++;
        }
        printf("평균 = %f\n", total/count);
        fclose(fp);
        return 0;
}
```

평균 = 58.575001

이진 화일 입출력 함수 : fread() 와 fwrite()

fread(*buffersize, n, *fp);
- buffer: 화일로부터 읽어 들인 데이터를 기억시킬 버퍼로의 포인터
- size : 한번에 읽어 들일 수 있는 데이터의 바이트 수
- n : size만큼 n 번 읽어 들이기 위해 지정하는 반복 read 회수
- fp : 대상이 되는 화일 포인터

fwrite(*buffer, size, n, *fp);
- buffer : 화일에 기록하고자 하는 데이터가 들어있는 버퍼로의 포인터
- size : 한번에 기록할 데이터의 바이트 수
- n : size만큼 n 번 쓰기 위해 지정하는 반복 write 회수
- fp : 대상이 되는 화일 포인터

- **텍스트 모드(fscanf, fprintf)에 비해 속도가 빠르다.**

- 내부적인 숫자 표현이나 제어문자에 대한 변환 과정이 없기 때문
- 많은 양의 데이터를 한꺼번에 읽거나 쓸 때 우수한 성능을 발휘한다.

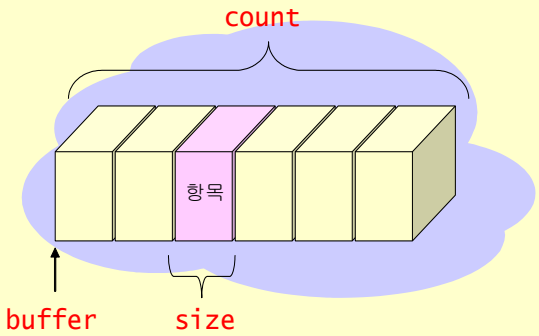
이진 화일의 생성

화일 모드	설명
"rb"	읽기 모드 + 이진 화일 모드
"wb"	쓰기 모드 + 이진 화일 모드
"ab"	추가 모드 + 이진 화일 모드
"rb+"	읽고 쓰기 모드 + 이진 화일 모드
"wb+"	쓰고 읽기 모드 + 이진 화일 모드

```
1.  int main(void)
2.  {
3.          FILE *fp = NULL;
4.
5.          fp = fopen("binary.txt", "rb");
6.
7.          if( fp == NULL )
8.                  printf("이진 화일 열기에 실패하였습니다.\n");
9.          else
10.                 printf("이진 화일 열기에 성공하였습니다.\n");
11.
12.         if( fp != NULL ) fclose(fp);
13. }
```

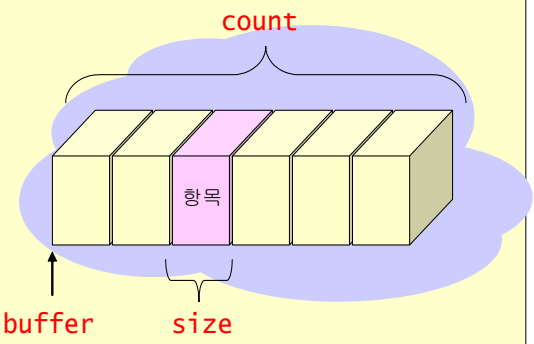
이진 화일 쓰기

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

2.  int main(void)
3.  {
4.          int buffer[] = { 10, 20, 30, 40, 50 };
5.          FILE *fp = NULL;
6.          size_t i, size, count;

7.          fp = fopen("binary.txt", "wb");
8.          if( fp == NULL )
9.          {
10.                 fprintf(stderr, "binary.txt 화일을 열 수 없습니다.");
11.                 exit(1);
12.         }

13.         size = sizeof(buffer[0]);
14.         count = sizeof(buffer) / sizeof(buffer[0]);

15.         i = fwrite(&buffer, size, count, fp);
16.         return 0;
17. }
```



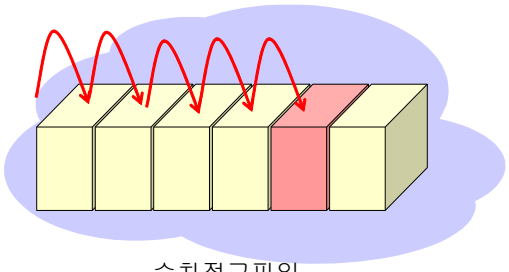
이진 화일 읽기

```
1.  #include <stdio.h>
2.  #define SIZE 1000

3.  int main(void)
4.  {
5.          float buffer[SIZE];
6.          FILE *fp = NULL;
7.          size_t size;

8.          fp = fopen("binary.txt", "rb");
9.          if( fp == NULL )
10.         {
11.                 fprintf(stderr, "binary.txt 화일을 열 수 없습니다.");
12.                 exit(1);
13.         }
14.         size = fread( &buffer, sizeof(float), SIZE, fp);
15.         if( size != SIZE )
16.         {
17.                 fprintf(stderr, "읽기 동작 중 오류가 발생했습니다.\n");
18.         }
19.         fclose(fp);

20.         return 0;
21. }
```

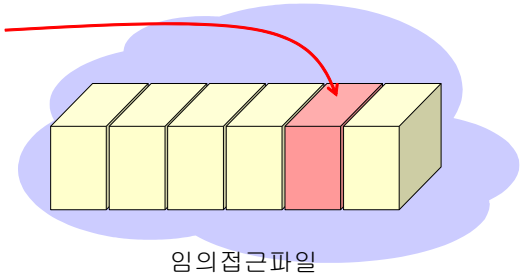


임의 접근 화일

- **순차 접근(sequential access)** 방법
  - 데이터를 화일의 처음부터 순차적으로 읽거나 기록하는 방법
- **임의 접근(random access)** 방법
  - 화일의 어느 위치에서든지 읽기와 쓰기가 가능한 방법



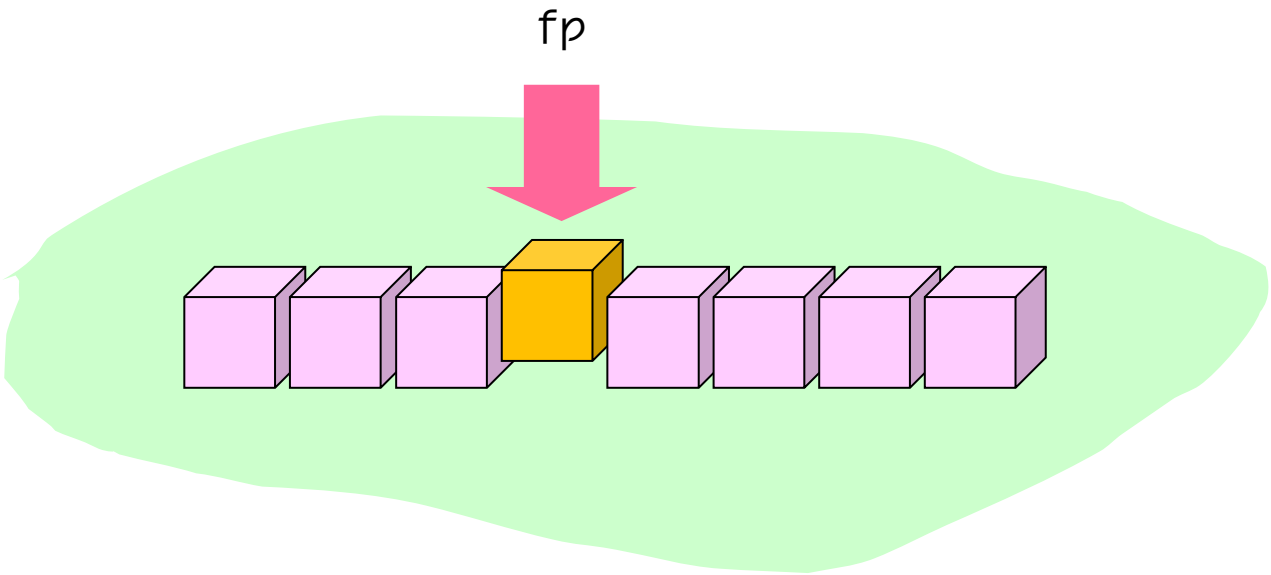
순차접근파일



임의접근파일

임의 접근 화일의 원리

- 화일포인터
  - 화일 위치 표시자
  - 읽기와 쓰기 동작이 현재 어떤 위치에서 이루어지는 지를 나타낸다.



- 강제적으로 화일 위치 표시자를 이동시키면 임의 접근이 가능

임의 접근 관련 함수

```
fseek(*fp, offset, origin);
```

origin	값	설명
SEEK_SET	0	화일의 시작
SEEK_CUR	1	현재 위치
SEEK_END	2	화일의 끝

```
fseek(fp, 0L, SEEK_SET);                    // 화일의 처음으로 이동
fseek(fp, 0L, SEEK_END);                    // 화일의 끝으로 이동
fseek(fp, 100L, SEEK_SET);                  // 화일의 처음에서 100바이트 이동
fseek(fp, 50L, SEEK_CUR);                   // 현재 위치에서 50바이트 이동
fseek(fp, -20L, SEEK_END);                  // 화일의 끝에서 20바이트 앞으로 이동
fseek(fp, sizeof(struct element), SEEK_SET);  // 구조체만큼 앞으로 이동
```

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

화일 위치 표시자의 현재 위치를 반환

fseek.c

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

#define SIZE 1000
void init_table(int table[], int size);

int main(void)
{
        int table[SIZE];
        int n, data;
        long pos;
        FILE *fp = NULL;

        // 배열을 초기화한다.
        init_table(table, SIZE);

        // 이진 화일을 쓰기 모드로 연다.
        if( (fp = fopen("sample.dat", "wb")) == NULL )
        {
                fprintf(stderr,"출력을 위한 화일을 열 수 없습니다.\n");
                exit(1);
        }

        // 배열을 이진 모드로 화일에 저장한다.
        fwrite(table, sizeof(int), SIZE, fp);
        fclose(fp);
```

```c
// 배열을 인덱스의 제곱으로 채운다.
void init_table(int table[], int size)
{
        int i;

        for(i = 0; i < size; i++)
                table[i] = i * i;
}
```

fseek.c(계속)

```
        // 이진 화일을 읽기 모드로 연다.
        if( (fp = fopen("sample.dat", "rb")) == NULL )
        {
                fprintf(stderr,"입력을 위한 화일을 열 수 없습니다.\n");
                exit(1);
        }
        // 사용자가 선택한 위치의 정수를 화일로부터 읽는다.
        while(1)
        {
                printf("화일에서의 위치를 입력하십시요(0에서 %d, 종료-1): ", SIZE - 1);
                scanf("%d", &n);
                if( n == -1 ) break
                pos = (long) n * sizeof(int);
                fseek(fp, pos, SEEK_SET);
                fread(&data, sizeof(int), 1, fp);
                printf("%d 위치의 값은 %d입니다.\n", n, data);
        }
        fclose(fp);
        return 0;
}
```

```
화일에서의 위치를 입력하십시요(0에서 999, 종료 -1): 3
3 위치의 값은 9입니다.
화일에서의 위치를 입력하십시요(0에서 999, 종료 -1): 9
9 위치의 값은 81입니다.
화일에서의 위치를 입력하십시요(0에서 999, 종료 -1): -1
```