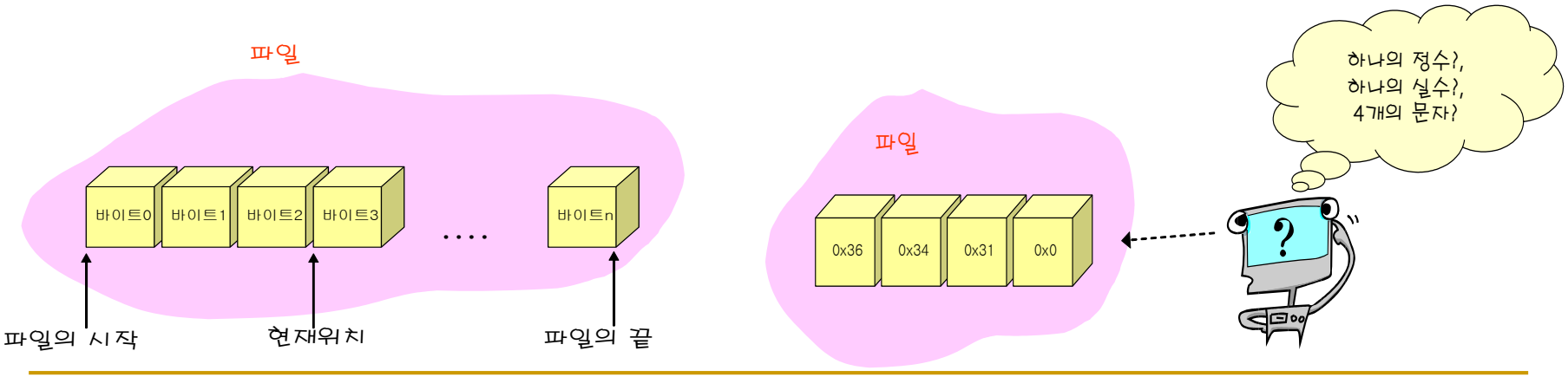
제 12 장 파일 입출력

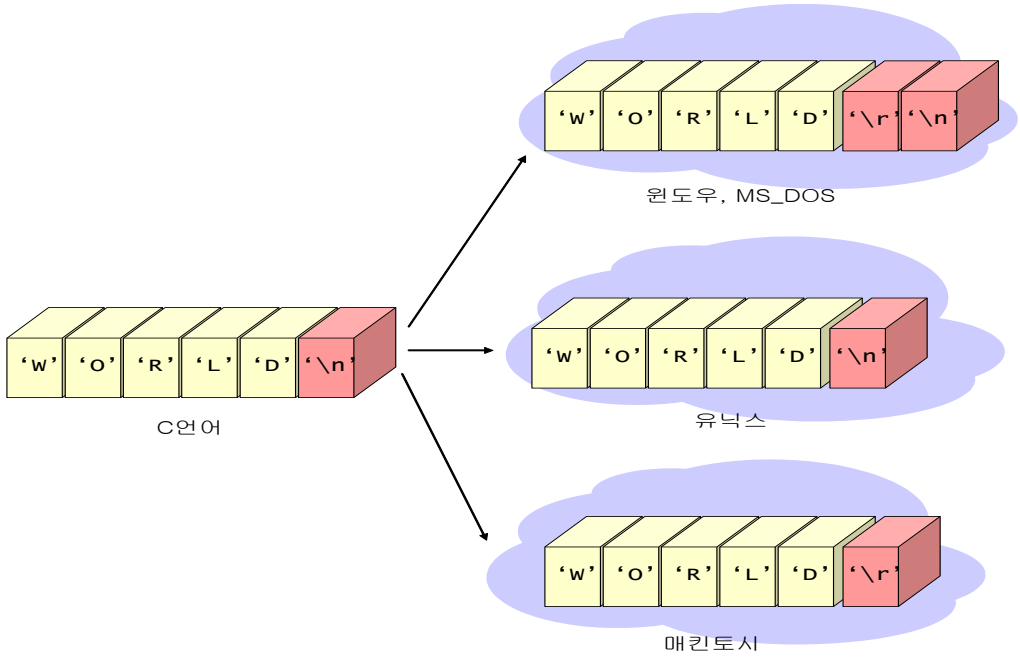
파일의 개념

- C에서의 파일은 일련의 연속된 바이트
- 모든 파일 데이터들은 결국은 바이트로 바뀌어서 파일에 저장
- 이들 바이트들을 어떻게 해석하느냐는 전적으로 프로그래머의 책임
  - 파일에 4개의 바이트가 들어 있을 때 이것을 int형의 정수 데이터로도 해석할 수 있고 아니면 float형 실수 데이터로도 해석할 수 있다.



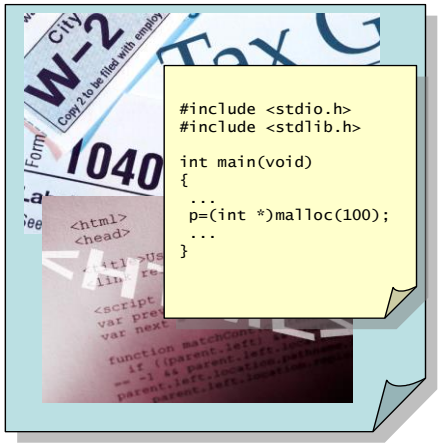
텍스트 파일(text file)

- 텍스트 파일은 사람이 읽을 수 있는 텍스트가 들어 있는 파일
  - (예) C 프로그램 소스 파일이나 메모장 파일
- 텍스트 파일은 연속적인 라인들로 구성

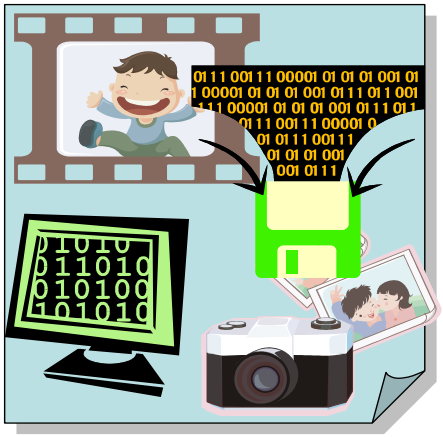


이진 파일(binary file)

- 이진 데이터가 직접 저장되어 있는 파일
- 이진 파일은 텍스트 파일과는 달리 라인들로 분리되지 않는다.
- 모든 데이터들은 문자열로 변환되지 않고 입출력
- 이진 파일은 특정 프로그램에 의해서만 판독이 가능
  - (예) C 프로그램 실행 파일, 사운드 파일, 이미지 파일



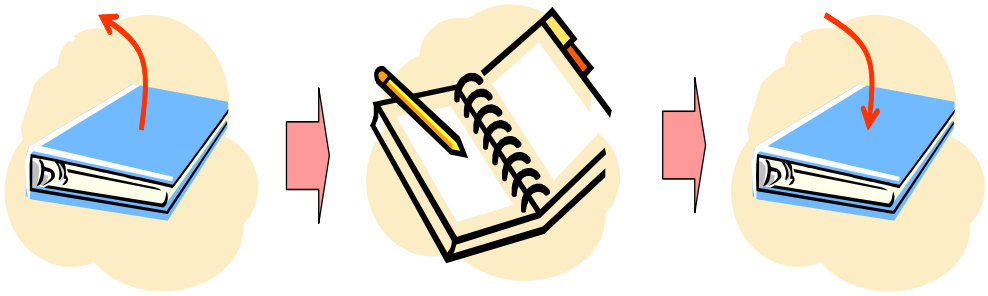
텍스트 파일: 문자로 구성된 파일



이진 파일: 데이터로 구성된 파일

파일 열기

- 파일을 다룰 때는 반드시 다음과 같은 순서를 지켜야 한다.



- 디스크 파일은 FILE 구조체를 이용하여 접근
- FILE 구조체를 가리키는 포인터를 파일 포인터(file pointer)

FILE 구조체

```
struct _iobuf {
        char *_ptr;
        int _cnt;
        char *_base;
        int _flag;
        int _file;
        int _charbuf;
        int _bufsize;
        char *_tmpfname;
} ;
typedef struct _iobuf FILE;
```

파일 열기

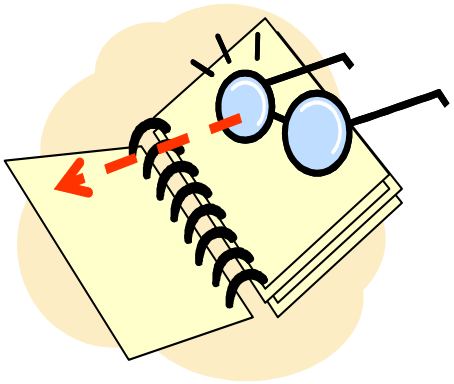
- 파일에서 데이터를 읽거나 쓸 수 있도록 모든 준비를 마치는 것

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

- 첫 번째 매개 변수인 name은 파일의 이름
- 두 번째 매개 변수인 mode는 파일을 여는 모드를 의미

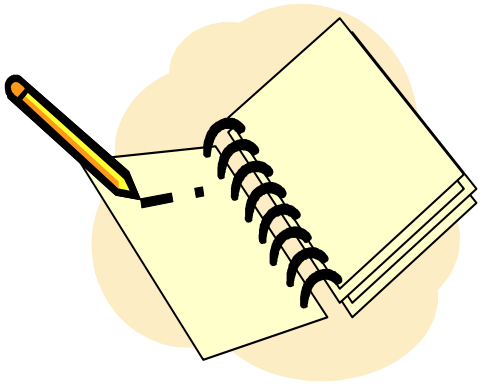
모드	설명
"r"	읽기 모드로 파일을 연다.
"w"	쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 이미 존재하면 기존의 내용이 지워진다.
"a"	추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 파일이 없으면 새로운 파일을 만든다.
"r+"	읽기와 쓰기 모드로 파일을 연다. 파일이 반드시 존재하여야 한다.
"w+"	읽기와 쓰기 모드로 파일을 생성한다. 만약 파일이 존재하지 않으면 파일이 생성된다. 파일이 존재하면 새 데이터가 기존 파일의 데이터를 덮어 쓰게 된다.
"a+"	읽기와 추가 모드로 파일을 연다. 만약 똑같은 이름의 기존의 파일이 있으면 데이터가 파일의 끝에 추가된다. 읽기는 어떤 위치에서나 가능하다. 파일이 없으면 새로운 파일을 만든다.
"b"	이진 파일 모드로 파일을 연다.

파일 모드



"r"

파일의 처음부터 읽는다.



"w"

파일의 처음부터 쓴다.
만약 파일이 존재하면 기존의 내용이 지워진다.

"a"

파일의 끝에 쓴다.
파일이 없으면 생성된다.

file_open.c

```
1.  // 파일 열기
2.  #include <stdio.h>
3.
4.  int main(void)
5.  {
6.          FILE *fp = NULL;
7.
8.          fp = fopen("sample.txt", "w");
9.
10.         if( fp == NULL )
11.                 printf("파일 열기 실패₩n");
12.         else
13.                 printf("파일 열기 성공₩n");
14.
15.         fclose(fp);
16.
17.         return 0;
18. }
```

```
파일 열기 성공
```

파일 닫기와 삭제

- 파일을 닫는 함수

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

- 파일을 삭제하는 함수

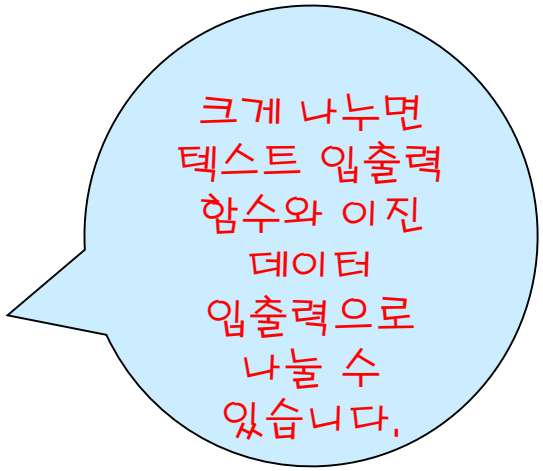
```
int remove(const char *path)
```

```
1.  #include <stdio.h>
2.
3.  int main( void )
4.  {
5.          if( remove( "sample.txt" ) == -1 )
6.                  printf( "sample.txt를 삭제할 수 없습니다.₩n" );
7.          else
8.                  printf( "sample.txt를 삭제하였습니다.₩n" );
9.
10.         return 0;
11. }
```

파일 입출력 함수

- **파일 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()



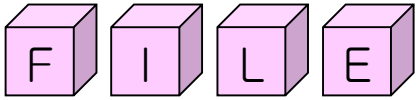
문자 단위 입출력

- **문자 입출력 함수**

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

파일 포인터

```
int fputc( int c, FILE *fp );
```



- **문자열 입출력 함수**

```
char *fgets( char *s, int n, FILE *fp );
```

문자열의 크기

```
int fputs( char *s, FILE *fp );
```

형식화된 출력(Formatted Output)

```
int fprintf( FILE *fp, const char *format, ...);
```

```
1.  int i = 23;
2.  float f = 1.2345;
3.  FILE *fp;
4.
5.  fp = fopen("sample.txt", "w");
6.
7.  if( fp != NULL )
8.          fprintf(fp, "%10d %16.3f", i, f);
9.
10. fclose(fp);
```

%d와 같은 특정한 형식을 지정하여 파일에 출력할 수 있습니다.

형식화된 입력

```
int fscanf( FILE *fp,  const char *format, ...);
```

```
1.  int i;
2.  float f;
3.  FILE *fp;
4.
5.  fp = fopen("sample.txt", "r");
6.
7.  if( fp != NULL )
8.          fscanf(fp, "%d %f", &i, &f);
9.
10. fclose(fp):
```

%d와 같은 특정한 형식을 지정하여 파일에 입력할 수 있습니다.

fcopy1.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  int main(void)
5.  {
6.          FILE *fp1, *fp2;
7.          char file1[100], file2[100];
8.          char buffer[100];
9.
10.         printf("원본 파일 이름:  ");
11.         scanf("%s", file1);
12.
13.         printf("복사 파일 이름:  ");
14.         scanf("%s", file2);
15.
16.         // 첫번째 파일을 읽기 모드로 연다.
17.         if( (fp1 = fopen(file1, "r")) == NULL )
18.         {
19.                 fprintf(stderr,"원본 파일 %s을 열 수 없습니다.₩n", file1);
20.                 exit(1);
21.         }
```

파일 복사 예제

```
22.             // 두번째 파일을 쓰기 모드로 연다.
23.             if( (fp2 = fopen(file2, "w")) == NULL )
24.             {
25.                             fprintf(stderr,"복사 파일 %s을 열 수 없습니다.₩n", file2);
26.                             exit(1);
27.             }
28.
29.             // 첫번째 파일을 두번째 파일로 복사한다.
30.             while( fgets(buffer, 100, fp1) != NULL )
31.                             fputs(buffer, fp2);
32.
33.             fclose(fp1);
34.             fclose(fp2);
35.
36.             return 0;
37.     }
```

원본 파일 이름: ***a.txt***
복사 파일 이름: ***b.txt***

search.c

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

3.  int main(void)
4.  {
5.          FILE *fp;
6.          char fname[128];
7.          char buffer[256];
8.          char word[256];
9.          int line_num = 0;

10.         printf("입력 파일 이름을 입력하시오:  ");
11.         scanf("%s", fname);

12.         printf("탐색할 단어를 입력하시오:  ");
13.         scanf("%s", word);
```

proverb.txt

```
A chain is only as strong as its weakest link
A change is as good as a rest
A fool and his money are soon parted
A friend in need is a friend indeed
A good beginning makes a good ending
A good man is hard to find
A house divided against itself cannot stand
A house is not a home
A journey of a thousand miles begins with a single step
A leopard cannot change its spots
A little knowledge is a dangerous thing
```

search.c

```
1.              // 파일을 읽기 모드로 연다.
2.              if( (fp = fopen(fname, "r")) == NULL )
3.              {
4.                              fprintf(stderr,"파일 %s을 열 수 없습니다.₩n", fname);
5.                              exit(1);
6.              }

7.              while( fgets(buffer, 256, fp) )
8.              {
9.                              line_num++;
10.                             if( strstr(buffer, word) )
11.                             {
12.                                             printf("%s: %d 단어 %s이 발견되었습니다.₩n", fname, line_num, word
    );
13.                             }
14.             }
15.             fclose(fp);

16.             return 0;
17. }
```

```
입력 파일 이름을 입력하시오: proverb.txt
탐색할 단어를 입력하시오: house
proverb.txt: 7 단어 house이 발견되었습니다.
proverb.txt: 8 단어 house이 발견되었습니다.
```

score3.c

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

3.  int main(void)
4.  {
5.          FILE *fp;
6.          char fname[100];
7.          int number, count = 0;
8.          char name[20];
9.          float score, total = 0.0;

10.         printf("성적 파일 이름을 입력하시오:  ");
11.         scanf("%s", fname);

12.         // 성적 파일을 쓰기 모드로 연다.
13.         if( (fp = fopen(fname, "w")) == NULL )
14.         {
15.                 fprintf(stderr,"성적 파일 %s을 열 수 없습니다.₩n", fname);
16.                 exit(1);
17.         }
```

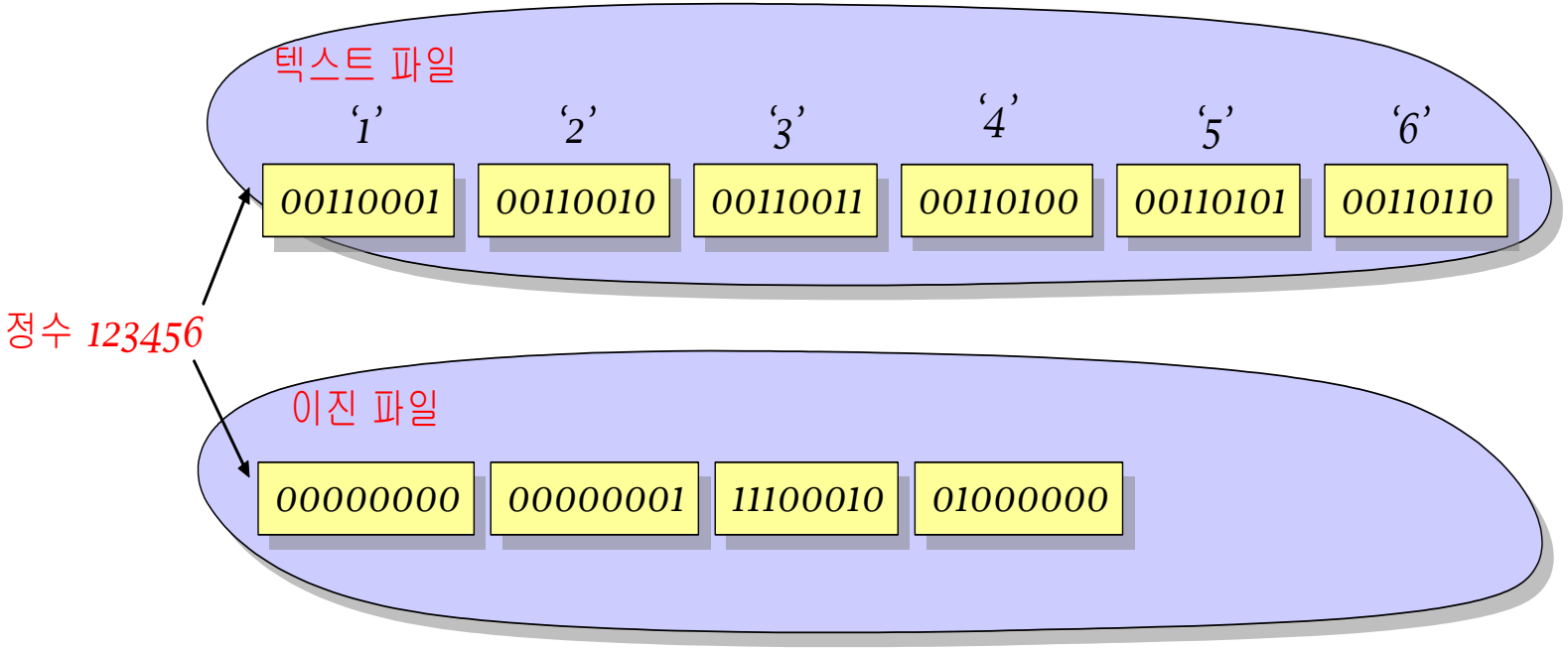
```
성적 파일 이름을 입력하시오: score.txt
학번, 이름, 성적을 입력하시오: (음수이면 종료) 1 KIM 90.2
학번, 이름, 성적을 입력하시오: (음수이면 종료) 2 PARK 30.5
학번, 이름, 성적을 입력하시오: (음수이면 종료) 3 MIN 56.8
학번, 이름, 성적을 입력하시오: (음수이면 종료)-1
평균 = 58.575001
```

score3.c

```
1.          // 사용자로부터 학번, 이름, 성적을 입력받아서 파일에 저장한다.
2.          while( 1 )
3.          {
4.                      printf("학번, 이름, 성적을 입력하시요: (음수이면 종료)");
5.                      scanf("%d", &number);
6.                      if( number < 0 ) break
7.                      scanf("%s %f", name, &score);
8.                      fprintf(fp, "%d %s %f₩n", number, name, score);
9.          }
10.         fclose(fp);
11.         // 성적 파일을 읽기 모드로 연다.
12.         if( (fp = fopen(fname, "r")) == NULL )
13.         {
14.                     fprintf(stderr,"성적 파일 %s을 열 수 없습니다.₩n", fname);
15.                     exit(1);
16.         }
17.         // 파일에서 성적을 읽어서 평균을 구한다.
18.         while( !feof( fp ) )
19.         {
20.                     fscanf(fp, "%d %s %f", &number, name, &score);
21.                     total += score;
22.                     count++;
23.         }
24.         printf("평균 = %f₩n", total/count);
25.         fclose(fp);
26.         return 0;
27. }
```

이진 파일 쓰기와 읽기

- 텍스트 파일과 이진 파일의 차이점
  - ***텍스트 파일***: 모든 데이터가 아스키 코드로 변환되어서 저장됨
  - ***이진 파일***: 컴퓨터에서 데이터를 표현하는 방식 그대로 저장



이진 파일의 생성

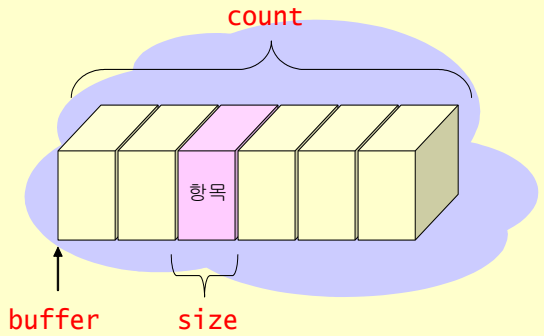
파일 모드	설명
"rb"	읽기 모드 + 이진 파일 모드
"wb"	쓰기 모드 + 이진 파일 모드
"ab"	추가 모드 + 이진 파일 모드
"rb+"	읽고 쓰기 모드 + 이진 파일 모드
"wb+"	쓰고 읽기 모드 + 이진 파일 모드

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

이진 파일 읽기

size_t fread(void *buffer, size_t size, size_t count, FILE *fp);

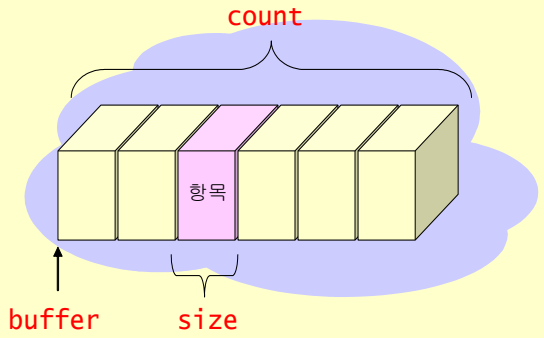
```
1.  #include <stdio.h>
2.  #define SIZE 1000

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 파일을 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. }
```



이진 파일 쓰기

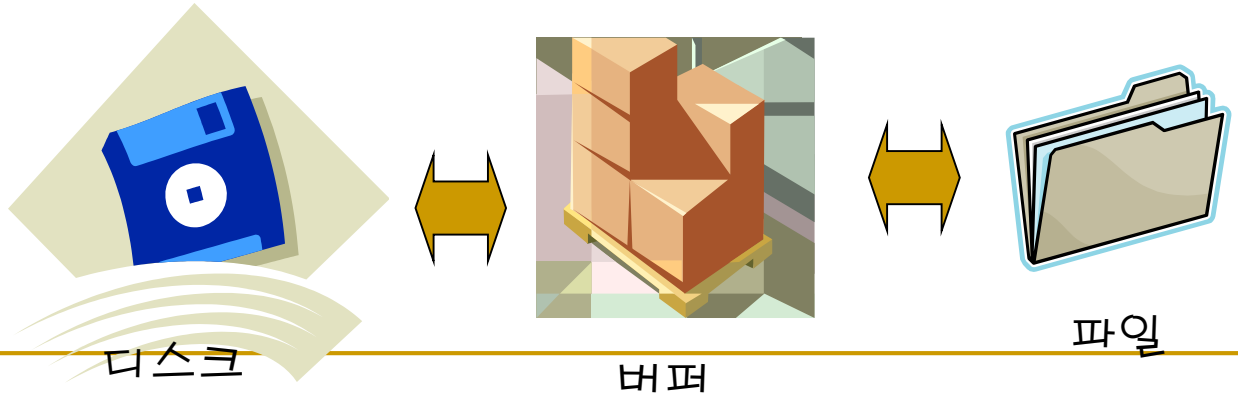
size_t fwrite(void *buffer, size_t size, size_t count, FILE *fp);

```
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 파일을 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. }
```



버퍼링

- fopen()을 사용하여 파일을 열면, 버퍼가 자동으로 만들어진다.
- 버퍼는 파일로부터 읽고 쓰는 데이터의 임시 저장 장소로 이용되는 메모리의 블록
- 디스크 드라이브는 블록 단위 장치이기 때문에 블록 단위로 입출력을 해야만 가장 효율적으로 동작
- 파일과 연결된 버퍼는 파일과 물리적인 디스크 사이의 인터페이스로 사용



binary_file.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #define SIZE 3

4.  struct student {
5.          int number;             // 학번
6.          char name[20];          // 이름
7.          double gpa;             // 평점
8.  };

9.  int main(void)
10. {
11.         struct student table[SIZE] = {
12.                 { 1, "Kim", 3.99 },
13.                 { 2, "Min", 2.68 },
14.                 { 3, "Lee", 4.01 }
15.         };
16.         struct student s;
17.         FILE *fp = NULL;
18.         int i;
19.         // 이진 파일을 쓰기 모드로 연다.
20.         if( (fp = fopen("student.dat", "wb")) == NULL )
21.         {
22.                 fprintf(stderr,"출력을 위한 파일을 열 수 없습니다.₩n");
23.                 exit(1);
24.         }

25.
```

binary_file.c

```
1.              // 배열을 파일에 저장한다.
2.              fwrite(table, sizeof(struct student), SIZE, fp);
3.              fclose(fp);

4.              // 이진 파일을 읽기 모드로 연다.
5.              if( (fp = fopen("student.dat", "rb")) == NULL )
6.              {
7.                              fprintf(stderr,"입력을 위한 파일을 열 수 없습니다.₩n");
8.                              exit(1);
9.              }

10.             for(i = 0;i < SIZE; i++)
11.             {
12.                             fread(&s, sizeof(struct student), 1, fp);
13.                             printf("학번 = %d, 이름 = %s, 평점 = %f₩n", s.number, s.name, s.gpa);
14.             }
15.             fclose(fp);

16.             return 0;
17.  }
```

학번 = 1, 이름 = Kim, 평점 = 3.990000
학번 = 2, 이름 = Min, 평점 = 2.680000
학번 = 3, 이름 = Lee, 평점 = 4.010000

fappend.c

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

3.  int main(void)
4.  {
5.          FILE *fp1, *fp2;
6.          char file1[100], file2[100];
7.          char buffer[1024];
8.          int count;

9.          printf("첫번째 파일 이름: ");
10.         scanf("%s", file1);

11.         printf("두번째 파일 이름:  ");
12.         scanf("%s", file2);

13.         // 첫번째 파일을 쓰기 모드로 연다.
14.         if( (fp1 = fopen(file1, "rb")) == NULL )
15.         {
16.                 fprintf(stderr,"입력을 위한 파일을 열 수 없습니다.₩n");
17.                 exit(1);
18.         }
```

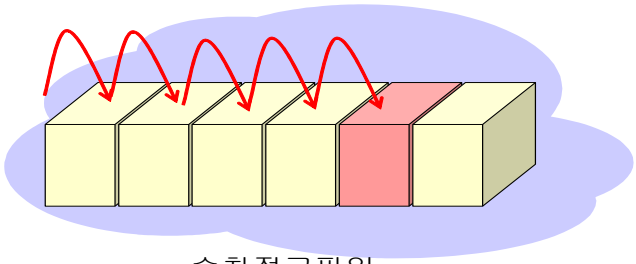
fappend.c

```
1.              // 두번째 파일을 추가 모드로 연다.
2.              if( (fp2 = fopen(file2, "ab")) == NULL )
3.              {
4.                              fprintf(stderr,"추가를 위한 파일을 열 수 없습니다.₩n");
5.                              exit(1);
6.              }

7.              // 첫번째 파일을 두번째 파일 끝에 추가한다.
8.              while((count = fread(buffer, sizeof(char), 1024, fp1)) > 0)
9.              {
10.                             fwrite(buffer, sizeof(char), count, fp2);
11.             }

12.             fclose(fp1);
13.             fclose(fp2);

14.             return 0;
15.  }
```

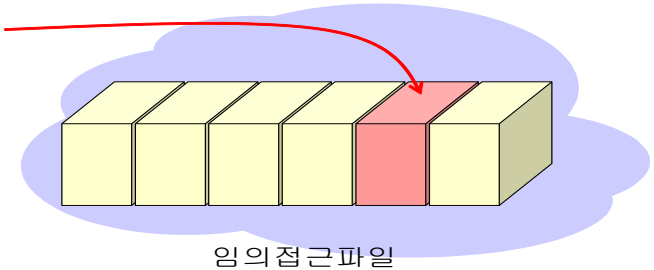
첫번째 파일 이름: a.dat
두번째 파일 이름: b.dat

임의 접근 파일

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



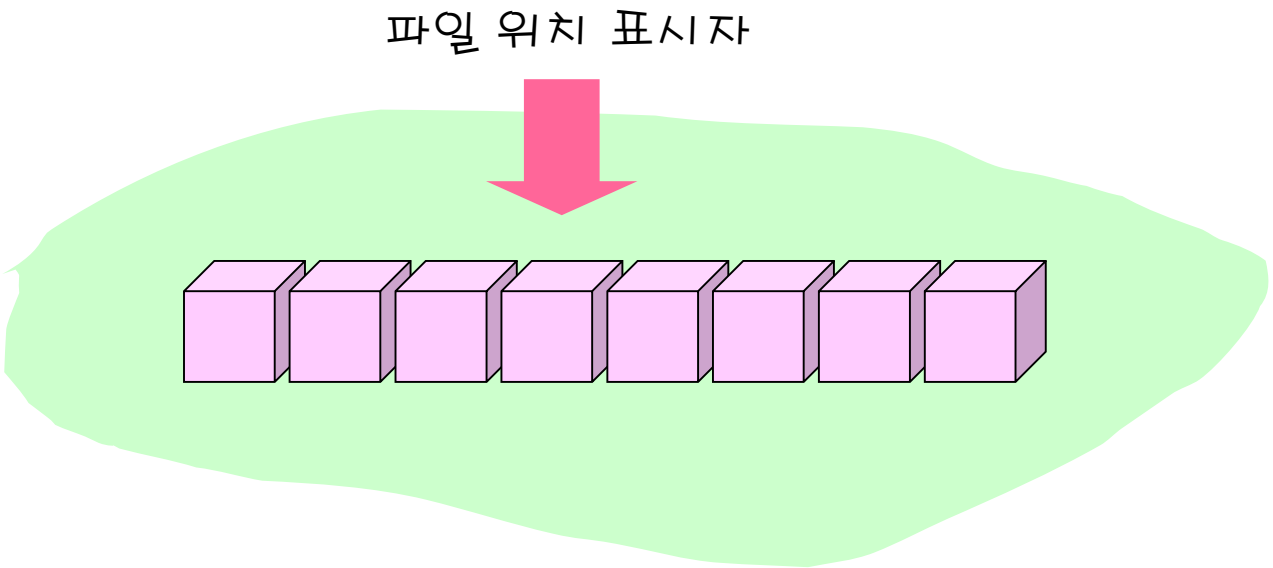
순차접근파일



임의접근파일

임의 접근 파일의 원리

- 파일 위치 표시자: 읽기와 쓰기 동작이 현재 어떤 위치에서 이루어지는 지를 나타낸다.



- 파일 위치 표시자를 특정 위치로 이동시키면 임의 접근이 가능

임의 접근 관련 함수

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

상수	값	설명
SEEK_SET	0	파일의 시작
SEEK_CUR	1	현재 위치
SEEK_END	2	파일의 끝

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

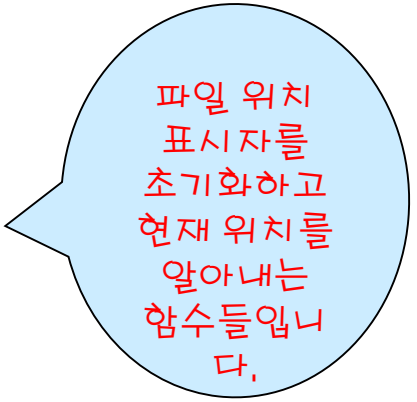
임의 접근 관련 함수

```
void rewind(FILE *fp);
```

파일 위치 표시자를 0으로 초기화

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

파일 위치 표시자의 현재 위치를 반환



fseek.c

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

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

5.  int main(void)
6.  {
7.          int table[SIZE];
8.          int n, data;
9.          long pos;
10.         FILE *fp = NULL;
11.
12.         // 배열을 초기화한다.
13.         init_table(table, SIZE);

14.         // 이진 파일을 쓰기 모드로 연다.
15.         if( (fp = fopen("sample.dat", "wb")) == NULL )
16.         {
17.                 fprintf(stderr,"출력을 위한 파일을 열 수 없습니다.₩n");
18.                 exit(1);
19.         }

20.         // 배열을 이진 모드로 파일에 저장한다.
21.         fwrite(table, sizeof(int), SIZE, fp);
22.         fclose(fp);
```

fappend.c

```
1.              // 이진 파일을 읽기 모드로 연다.
2.              if( (fp = fopen("sample.dat", "rb")) == NULL )
3.              {
4.                              fprintf(stderr,"입력을 위한 파일을 열 수 없습니다.₩n");
5.                              exit(1);
6.              }
7.              // 사용자가 선택한 위치의 정수를 파일로부터 읽는다.
8.              while(1)
9.              {
10.                             printf("파일에서의 위치를 입력하십시요(0에서 %d, 종료-1): ", SIZE - 1);
11.                             scanf("%d", &n);
12.                             if( n == -1 ) break
13.                             pos = (long) n * sizeof(int);
14.                             fseek(fp, pos, SEEK_SET);
15.                             fread(&data, sizeof(int), 1, fp);
16.                             printf("%d 위치의 값은 %d입니다.₩n", n, data);
17.             }
18.             fclose(fp);
19.             return 0;
20.     }
21.     // 배열을 인덱스의 제곱으로 채운다.
22.     void init_table(int table[], int size)
23.     {
24.             int i;

25.             for(i = 0; i < size; i++)
26.                             table[i] = i * i;
27.     }
```

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