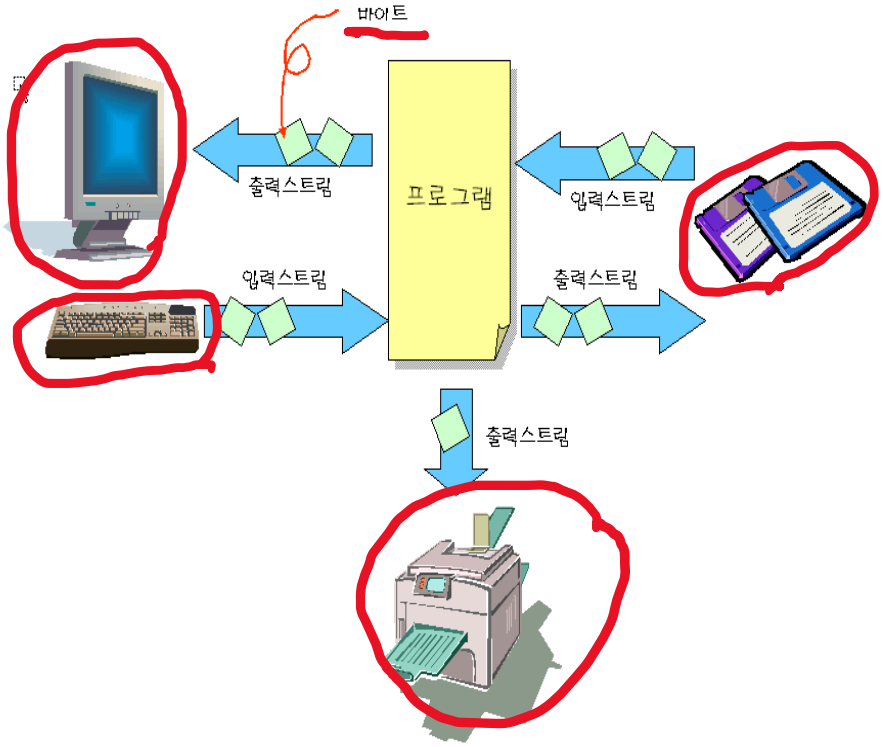
파일 입출력 Part 1

Stream의 개념

Stream 정의

- 스트림(stream): 바이트의 흐름
  - 모든 입, 출력 및 장치들을 바이트(byte)들의 흐름으로 간주
- 스트림 종류
  - 표준 입출력 스트림
    - 시스템에서 묵시적으로 설정해 놓은 입출력 스트림
    - stdin, stdout, stderr
  - 프로그래머 생성 입출력 스트림
    - 프로그래머가 지정하여 생성하는 입출력 스트림
    - FILE 구조체 기반의 파일 입출력 스트림
    - Socket 기반의 네트워크 입출력 스트림

Stream 개념



표준 입출력 스트림

- stdin
  - 표준 입력 스트림
    - 표준 입력 장치를 키보드로 지정하여, 키보드로부터의 입력을 “stdin” 이라고 하는 시스템에 의해 생성된 입력 스트림으로부터 데이터를 수신하는 개념으로 변환
    - scanf, getchar 계열의 입력 라이브러리가 사용
- stdout
  - 표준 출력 스트림
    - 표준 출력 장치를 모니터(화면)로 지정하여, 모니터로의 출력을 “stdout”이라고 하는 시스템에 의해 생성된 출력 스트림으로 데이터를 싣는 개념으로 변환
    - printf, putchar 계열의 출력 라이브러리가 사용
- stderr
  - 표준 오류 스트림으로 출력 스트림과 동일한 개념으로 사용

표준입출력 스트림

- stdin : 키보드의 추상화
- stdout : 모니터의 추상화
- stderr : 오류 메시지 출력 스트림으로 stdout과 동일

- 표준입출력스트림을 포함하여 총 생성 가능한 스트림의 개수는 512개
- formatted stream
  - printf, scanf 계열
- Unformatted stream
  - Putchar, getchar 계열

표준입출력 스트림 개념도

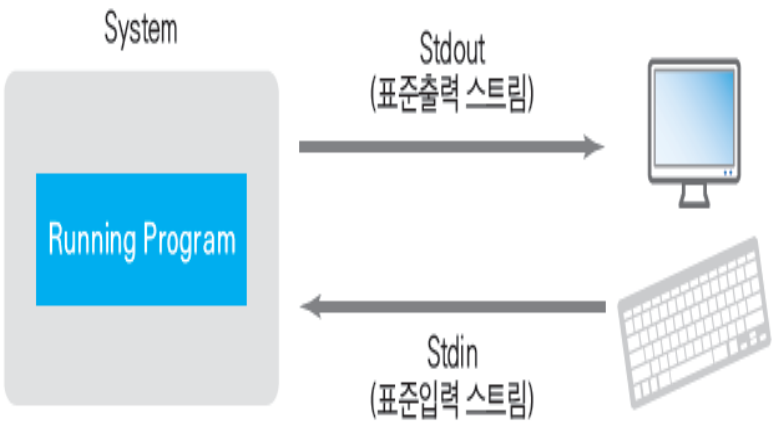
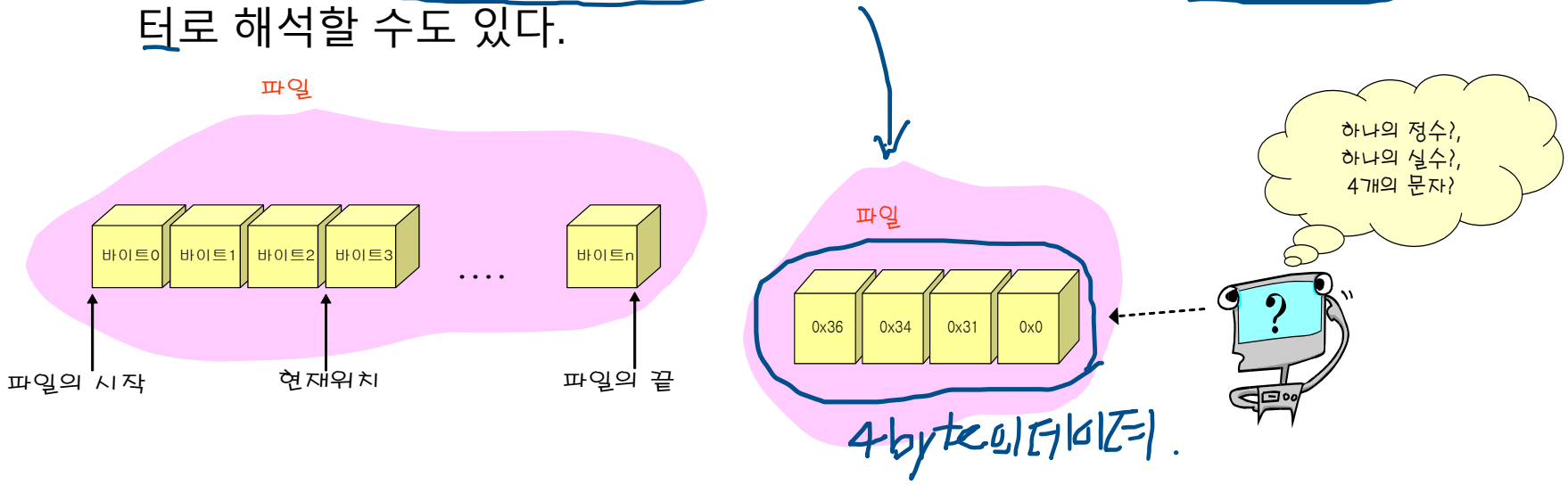


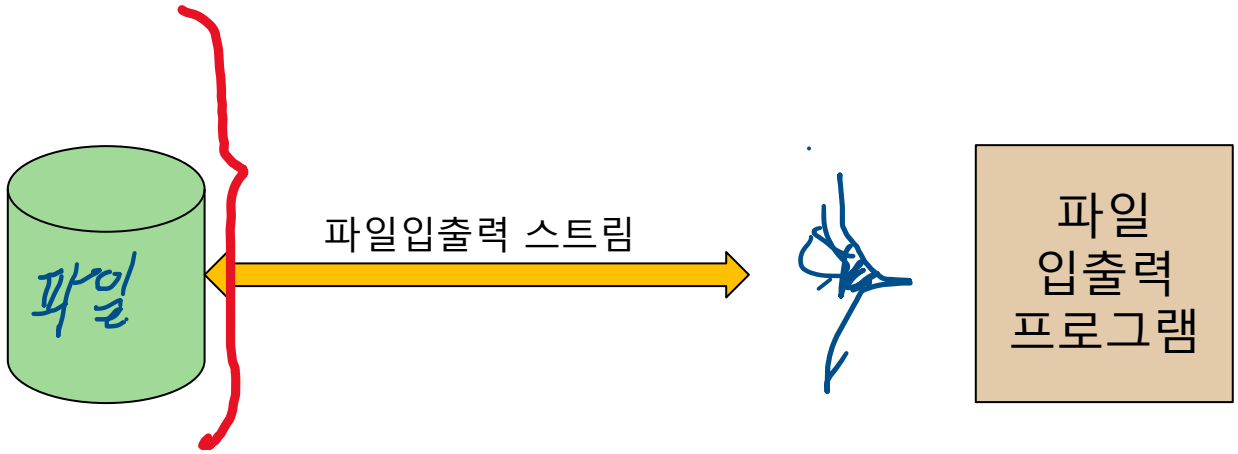
그림 12-1 표준 입출력 스트림

파일 입출력 스트림

- C 언어에서의 파일은 스트림(Stream)으로 간주
- 파일 데이터는 결국은 바이트 단위로 파일에 저장하고 바이트 단위로 읽혀져 메모리에 탑재된다
- 이들 바이트들에 대한 해석은 전적으로 프로그래머에 의존한다
  - 파일에 4개의 바이트가 들어 있을 때 이것을 int형의 정수 데이터로 해석할 수도 있고, float형 실수 데이터로도 해석할 수 있으며 4개의 문자 데이터로 해석할 수도 있다.

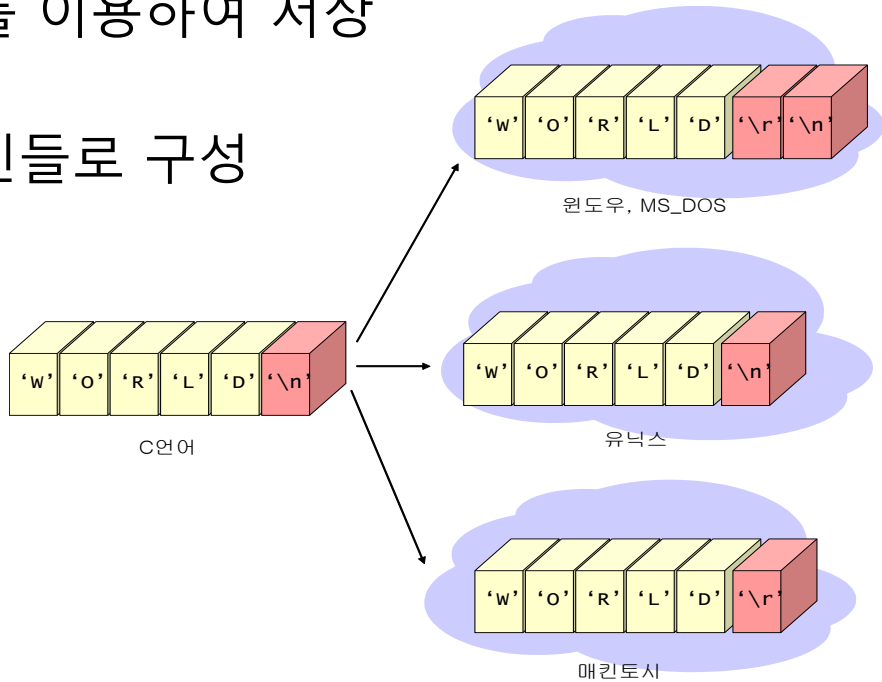


- 파일을 입출력 하기 위해서는 먼저 스트림을 생성하여야 한다
- 파일 구조체를 사용 (FILE)
- 과정
  - **파일 스트림 생성 : fopen()**
  - **파일 데이터 처리 : read/write/search**
  - **파일 스트림 제거 : fclose()**

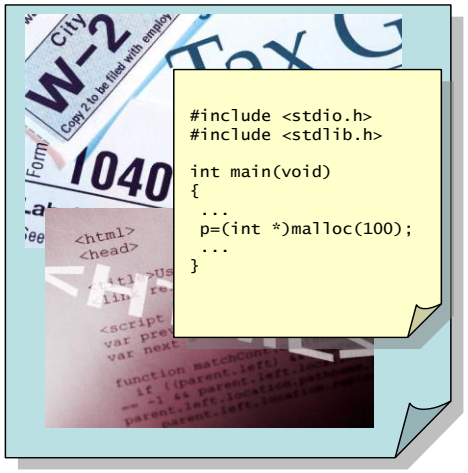


파일 스트림의 종류

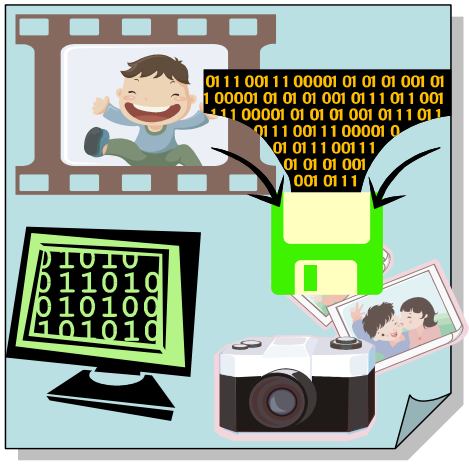
- **텍스트 파일스트림 vs. 2진 파일스트림**
- 텍스트 파일스트림
  - 사람이 인지할 수 있는 텍스트가 저장된 파일
    - (예) C 프로그램 소스 파일이나 메모장 파일
  - 텍스트 파일은 아스키 코드를 이용하여 저장
    - 문자 변환이 일어날 수 있음
  - 텍스트 파일은 연속적인 라인들로 구성



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



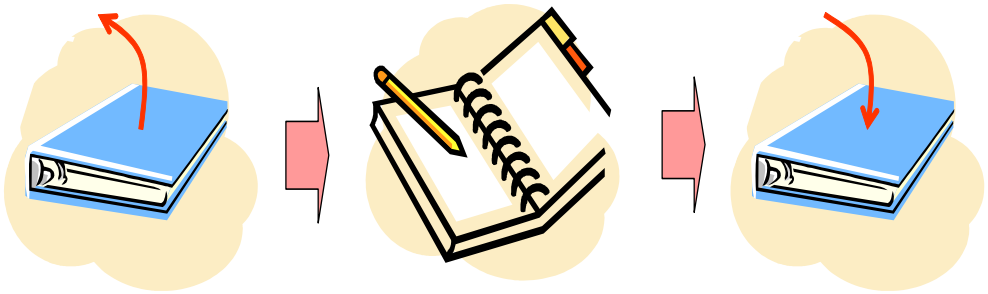
텍스트 파일: 문자로 구성된 파일



이진 파일: 데이터로 구성된 파일

파일 열기 : fopen( )

- 파일 처리는 반드시 다음의 순서를 지켜야 한다.

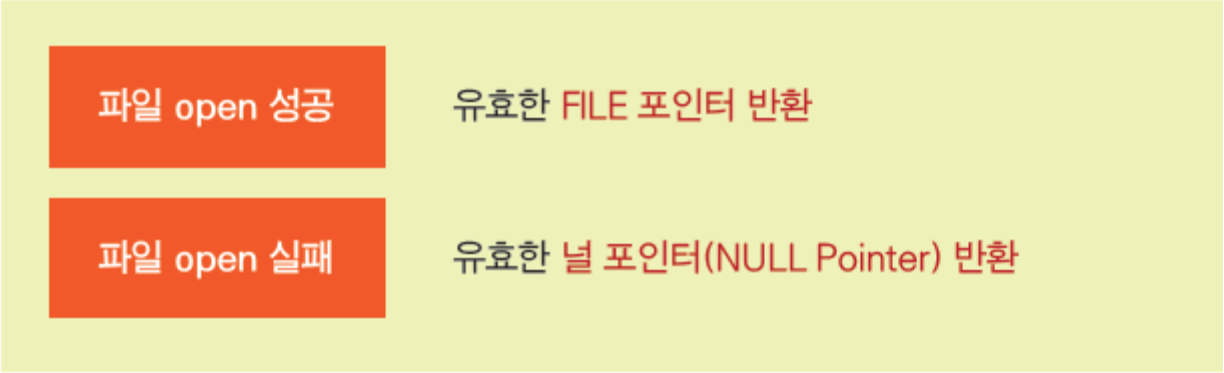


- 파일은 FILE 구조체를 이용하여 스트림 생성한 후 접근
  - 파일에서 데이터를 읽거나 쓸 수 있도록 모든 준비
  - 파일에 관한 모든 정보를 FILE 구조체에 저장
- FILE 구조체를 가리키는 포인터를 파일 포인터(file pointer)

fopen() 함수

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

- 매개 변수 name은 **파일명** 문자열을 가리키는 포인터
- 매개 변수 mode는 파일을 여는 **모드** 문자열을 가리키는 포인터
  - 텍스트 모드 및 이진 모드
- fopen()을 성공적으로 마치면 FILE 구조체를 가리키는 파일 포인터를 반환, 실패하면 NULL 포인터를 반환



fopen( )의 모드

모드	처 리	파일이 없는 경우	파일이 있는 경우
"r"	텍스트 파일 읽기(read)	NULL 반환	정상 처리
"w"	텍스트 파일 저장(write) 하기	새로운 파일 생성	이전 내용 삭제
"a"	텍스트 파일 추가(append)	새로운 파일 생성	이전 내용 뒤에 추가
"r+"	텍스트 파일 읽고 저장하기	NULL 반환	정상 처리
"w+"	텍스트 파일 읽고 저장하기	새로운 파일 생성	이전 내용 삭제
"a+"	텍스트 파일 추가로 읽고 저장	새로운 파일 생성	이전 내용 뒤에 추가
"rb"	2진 파일 읽기	NULL 반환	정상처리
"wb"	2진 파일 저장 하기	새로운 파일 생성	이전 내용 삭제
"ab"	2진 파일 추가	새로운 파일 생성	이전의 내용 뒤에 추가
"r+b"	2진 파일 읽고 저장하기	NULL 반환	정상 처리
"w+b"	2진 파일 읽고 저장하기	새로운 파일 생성	이전 내용 삭제
"a+b"	2진 파일 추가로 읽고 저장하기	새로운 파일 생성	이전 내용 뒤에 추가

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 *fp );
```

- 파일을 삭제하는 함수

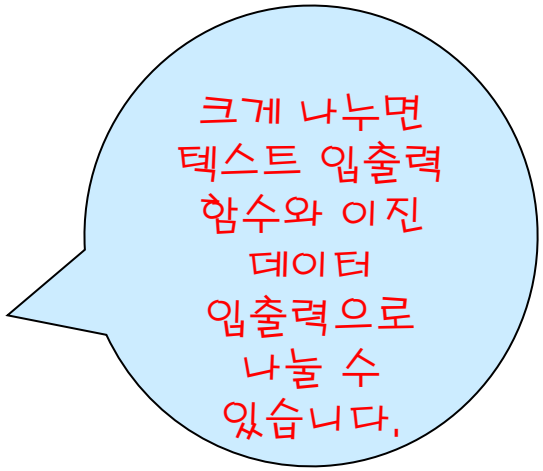
```
int remove(const char *path)
```

```
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()



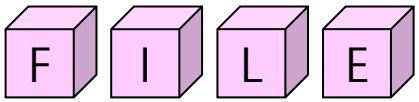
문자 단위 입출력

- 문자 입출력 함수

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

파일 포인터

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

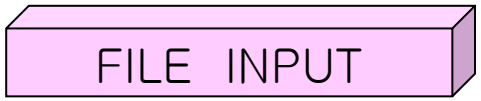


- 문자열 입출력 함수

```
char *fgets( char *buffer, int buf_size, FILE *fp );
```

문자열의 크기

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



문자입출력

A.txt 를 A.bak 파일로 복사한다.

```
C':\>  my_copy.exe A.txt A.bak
```

문자 입출력, 13_1

```
4   int main(int argc, char *argv[])
5   {
6       FILE *in, *out;  //파일 포인터 선언
7       char ch;
8
9       if(argc ! =3)
10      {
11          printf("Usage CH12 <filename1> <filename2> \n");
12          exit(1);
13      }
14
15      in = fopen(argv[1], "r");  <------ 파일 열기, 복사할 파일
16      if(in == NULL)
17      {
18          printf("%s file open error. \n", argv[1]);
19          exit(1);
20      }
```

문자 입출력, 13_1

```
22      if((out = fopen(argv[2], "w")) == NULL)  <------ 파일 열기, 복사 받을 파일
23      {
24          printf("%s file open error. \n", argv[2]);
25          exit(1);
26      }
```

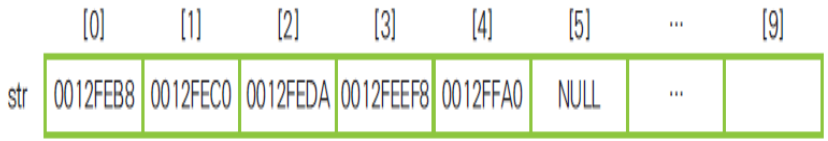
문자 입출력, 13_1

```
28      while((ch = fgetc(in)) != EOF)  <------ 파일 끝까지 한 문자씩 읽음
29          fputc(ch, out);  <------ 읽어 들인 문자를 out 스트림에 출력한다
30
31      printf("%s가 %s 로 복사됨...\n", argv[1], argv[2]);
32      fclose(in);
33      fclose(out);
34      return 0;
35  }
```

문자열 입출력

예제 포인터 배열이 참조하는 주소에 저장된 문자열을 파일에 저장하고 읽어보자.

포인터 배열 str이 갖는 정보는 다음과 같다.



0012FFB8	kingdom\n\0
0012FFC0	king\n\0
0012FFDA	queen\n\0
0012FFF8	prince\n\0
0012FFA0	princess\n\0

실행결과

```
kingdom
king
queen
prince
princess
```

문자열 입출력, 13_2

```
4   int main(void)
5   {
6       char *str[10] = {"kingdom\n", "king\n", "queen\n",
7                        "prince\n", "princess\n", NULL };
8       FILE *fp;
9       char tmp[20];
10      int i = 0;
```

문자열 입출력, 12_2

```
12      if((fp = fopen("DATA2.txt", "wt")) == NULL) <------ 스트림 연결
13      {
14          printf("file open error. \n");
15          exit(1);
16      }
17
18      while(str[i])
19      {
20          fputs(str[i], fp); <------ 파일 저장
21          i++;
22      }
```

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.

파일복사예

```
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.         }
```

```
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***

탐색예

```
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
```

```
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.
```

성적평균 구하기 예

```
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***

```
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. }
```