15 파일 처리

함수 fopen()

- 파일을 만들기 위해서는 함수 fopen()을 이용
- 함수 fopen()의 함수 원형은 다음과 같으며 헤더 파일 stdio.h 파일에 정의

```
FILE * fopen( const char *, const char * );
```

첫 번째 인자는 처리하려는 파일 이름을 지정하는 인자이다.

두 번째 인자는 처리하려는 파일 처리의 종류를 지정하는 인자이다.

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

- 함수 fopen()은 두 개의 문자열 전달인자를 이용, 반환 값은 포인터 값인 FILE *

파일 열기

- 함수 fopen()에서 첫 번째 문자열은 처리하려는 파일 이름이고, 두 번째 문자열은 파일 처리 종류(모드)
- 다음 소스는 파일 "basic.txt"을 여는 모듈로서, 파일에 자료를 쓰기 위해 파일을 열므로 모드 값을 "w"로 기술

```
FILE *fp;
char fname[] = "basic.txt";
if ((fp = fopen(fname, "w")) == NULL) {
   printf( "파일이 열리지 않습니다.\n" );
}
…

fclose(fp);
```

- 조건문 if를 위와 같이 함수 fopen()과 함께 이용하면 파일 열기에 실패할 경우 문장 "파일이 열리지 않습니다.\n"을 출력
- 파일 처리가 모두 끝났으면 파일 포인터 fp를 인자로 함수 fclose(fp)를 호출하여 반드시 파일을 닫도록
- 함수 fopen()의 두 번째 인자는 파일 처리 종류(모드)
  - "r", "w", "a", "r+", "w+", "a+"의 종류

파일 처리 모드

❖ 파일 처리 모드 종류 의미

모드	의 미
r	읽기(read) 모드로 파일을 연다. 파일이 없으면 에러가 발생한다.
w	쓰기(write) 모드로 파일을 연다. 파일이 없으면 새로 만들고, 기존의 파일이 있으면 그 이전의 내용은 없어지고 파일의 처음부터 쓴다. 이 모드로는 파일 내용을 읽을 수 없다.
a	추가 쓰기(append) 모드로 파일을 연다. 파일이 없으면 새로 만들고, 기존의 파일이 있으면 그 파일의 가장 뒤부터 파일에 추가한다.
r+	읽기(read)와 쓰기(write) 모드로 파일을 연다. 파일이 없으면 에러가 발생한다.
w+	읽기와 쓰기(write) 모드로 파일을 연다. 파일이 없으면 새로 만들고, 기존의 파일이 있으면 그 이전의 내용은 없어지고 파일의 처음부터 쓴다.
a+	추가 쓰기(append) 모드로 파일을 연다. 파일이 없으면 새로 만들고, 기존의 파일이 있으면 그 파일의 가장 뒤부터 파일에 추가한다. 파일의 어느 곳이나 읽기는 가능하나 쓰기는 파일 끝에 추가적으로만 가능하다.

표 17.1 파일 열기 함수 fopen( )의 모드 종류

함수 fprintf, fscanf

- 파일에 자료를 쓰거나 읽기 위하여 함수 fprintf()와 fscanf()를 이용
  - 이 함수를 이용하기 위해서는 헤더 파일 stdio.h 파일을 포함
- 함수 fprintf()와 fscanf()의 함수 원형

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

- 위 함수의 첫 번째 인자는 입출력에 이용될 파일이고, 두 번째 인자는 입출력되는 문자열이며, 다음 인자들은 입출력될 변수 목록
  - 함수 원형에서 기호 …은 여러 인자가 계속됨을 의미
- 함수 fprintf()와 fscanf()를 표준 입출력에도 이용 가능
  - 즉 함수 fprintf()와 fscanf()의 첫 번째 인자에 각각 stdin 또는 stdout를 이용하면 표준 입력, 표준 출력으로 이용이 가능
- 기호 상수 stdin, stdout은 stderr과 함께 헤더 파일 stdio.h에 정의되어 있는 값으로 각각 표준입력, 표준출력, 표준에러를 의미

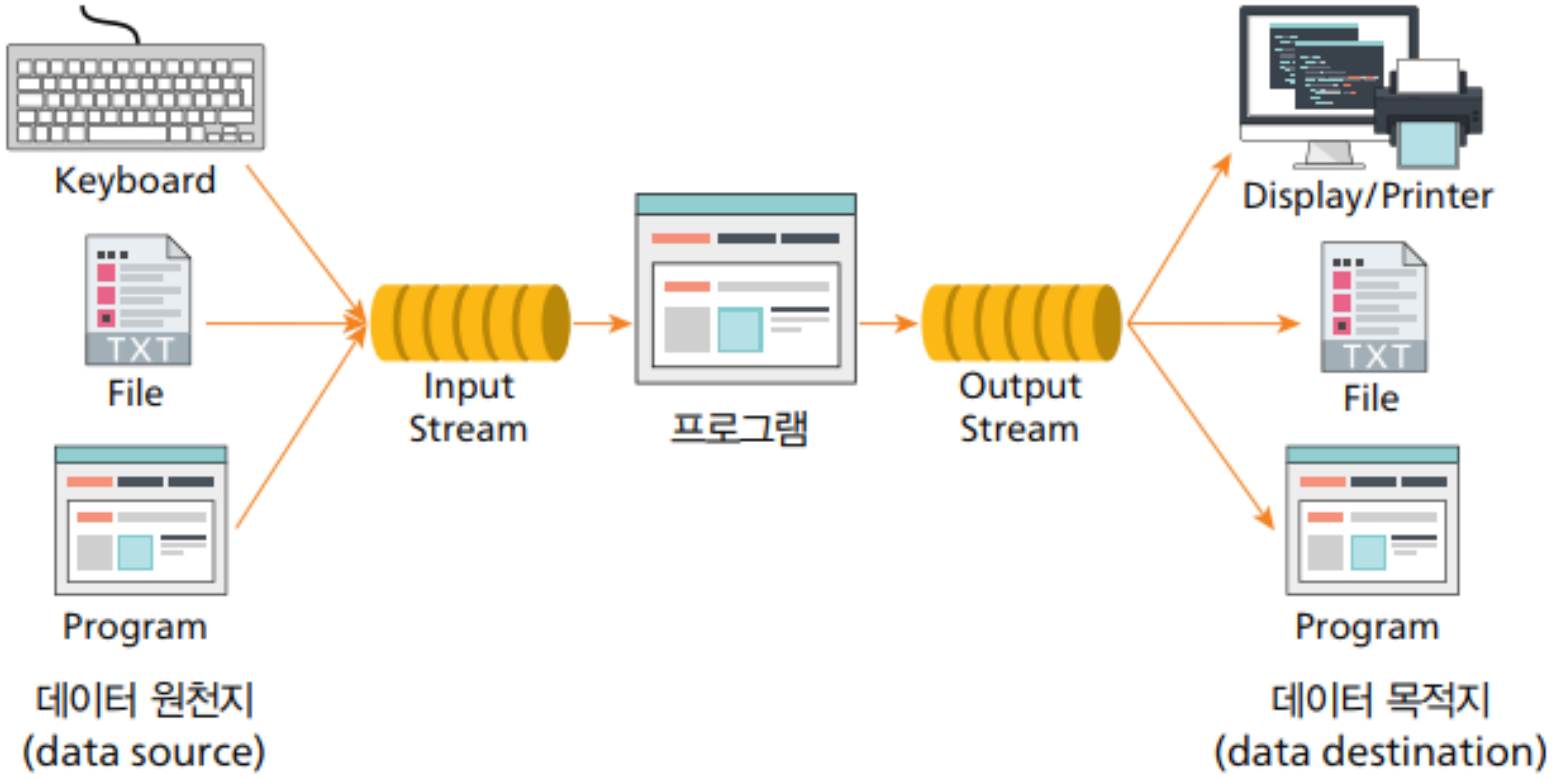
```
#define stdin  (&_iob[0])
#define stdout (&_iob[1])
#define stderr (&_iob[2])
```

표준 파일	키워드	장치(device)
표준입력	stdin	키보드
표준출력	stdout	모니터 화면
표준에러	stderr	모니터 화면

입출력 스트림

❖ 입출력 스트림(io stream)

- 입출력 시 이동 통로
- 표준입력 스트림: 키보드 → 프로그램
- 표준출력 스트림: 프로그램 → 모니터의 콘솔



함수 fgets()와 fputs()

❖ 함수 fgets()와 fputs()

- 파일에 문자열을 입출력하는 함수로 fgets()와 fputs()
  - 이 함수도 헤더파일 stdio.h 파일에 다음과 같은 함수원형으로 정의
- 함수 fgets()는 문자열을 개행문자(₩n)까지 읽어 개행문자도 함께 입력 문자열에 저장
- 마찬가지로 함수 fputs()는 문자열을 그대로 출력

❖ 함수 fgets() 인자

- 첫 번째 인자는 문자열이 저장될 문자 포인터이고,
- 두 번째 인자는 입력할 문자의 최대 수이며,
- 세 번째 인자는 입력 문자열이 저장될 파일

❖ 함수 fputs() 인자

- 첫 번째 인자는 출력될 문자열이 저장된 문자 포인터이고,
- 두 번째 인자는 문자열이 출력되는 파일

```
char * fgets(char *, int, FILE *);
int  fputs(char *, FILE *);
```

함수 fgetc()와 fputc()

- 문자 하나를 파일에 입출력하는 함수로 fgetc()와 fputc()를 제공
  - 이 함수의 원형은 헤더파일 stdio.h에 정의
- 이 함수들은 문자 하나의 입출력의 대상인 파일 포인터를 인자로 이용

```
int  fgetc(FILE *);
int  fputc(int, FILE *);
```

- 이 함수와 같은 함수로 getc()와 putc()도 이용 가능

```
int  getc(FILE *);
int  putc(int, FILE *);
```

- 문자의 표준 입출력에 이용되는 getchar()와 putchar()는 다음과 같이 함수 getc()와 putc()를 이용한 매크로

```
#define getchar()       getc(stdin)
#define putchar(_c)      putc((_c),stdout)
```

- 함수 fgetc()와 fputc()는 getc()와 putc()와 그 기능은 동일하며, fgetc()와 fputc()는 함수이고, getc()와 putc()는 매크로

함수 feof()와 ferror()

❖ 함수 feof()

- 파일의 내부 포인터 위치가 파일의 끝(EOF)인지를 검사하는데 필요한 함수
  - 이 함수는 헤더파일 stdio.h에 다음 함수 원형으로 정의

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

- 파일의 위치가 파일의 마지막(end of file)인지를 검사하여, 파일의 마지막이면 0이 아닌 값 (true) 을, 파일의 마지막이 아니면 0 (false) 을 반환
  - 그러므로 표준입력에서 계속적으로 입력을 받는 구문으로 다음을 이용 가능

```
while (!feof(stdin)) {
  …
}
```

❖ 함수 ferror()

- 파일 처리에서 오류가 발생했는지 검사하는 함수
  - 함수의 원형은 헤더파일 stdio.h에 정의

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

- 이전 파일 처리에서 오류가 발생하면 0이 아닌 값 (true)을 발생하고, 오류가 발생하지 않으면 0 (false) 을 반환

파일의 끝을 검사하기 1

❖ 한 글자씩 읽어서 **EOF** 와 검사하기

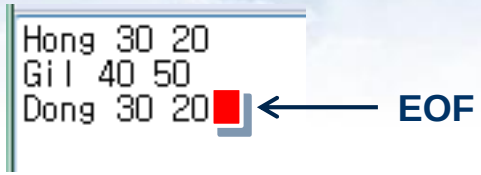
```
void eg11()
{
    FILE *fp;
    char ch;

    fp = fopen("test.cpp","r");
    if ( fp == NULL) {
        printf("Cannot find the file₩n");
        exit(0);
    }

    while( (ch = fgetc(fp)) != EOF )
        putchar(ch);

}
```

파일의 끝을 검사하기 2

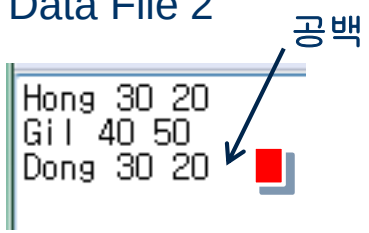
❖ feof 함수를 이용하여 검사하기

```
void eg13()
{
    FILE *fp;
    char name[30];
    int score1, score2;

    fp = fopen("data.txt","r");
    if ( fp == NULL) {
        printf("Cannot find the file₩n");
        exit(0);
    }

    while( !feof(fp) )
    {
        fscanf(fp, "%s %d %d", name, &score1, &score2);
        printf("%s %d %d₩n", name, score1, score2);
    }

    fclose(fp);
}
```

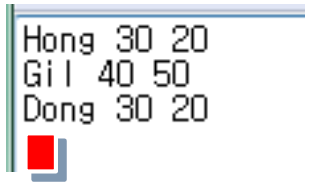
❖ Data File 1



❖ Data File 2



❖ Data File 3



파일의 끝을 검사하기 3

❖ **fscanf()** 함수를 이용하여 검사하기

```
void eg14( )
{
    FILE *fp;
    char name[30];
    int score1, score2;

    fp = fopen("data.txt","r");
    if ( fp == NULL) {
        printf("Cannot find the file₩n");
        exit(0);
    }

//  while( fscanf(fp, "%s %d %d", name, &score1, &score2) != EOF )
    while( fscanf(fp, "%s", name) != EOF )
    {
        fscanf(fp, "%d %d", &score1, &score2);
        printf("%s %d %d₩n", name, score1, score2);
    }

    fclose(fp);
}
```

파일 내용을 표준출력으로 그대로 출력

Prj05 05flist.c 지정한 파일 "05flist.c"의 내용을 행 번호를 붙여 표준출력으로 그대로 출력하는 프로그램 난이도: ★★

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int main(void)
05 {
06    FILE* f;
07    if (fopen_s(&f, "05flist.c", "r") != 0) //읽기 모드로 파일 열기
08    //if ( (f = fopen("05flist.c", "r")) == NULL )
09    {
10       printf("파일이 열리지 않습니다.\n");
11       exit(1);
12    }
13
14    int ch, cnt = 0; //문자를 저장할 ch, 행번호를 저장할 cnt
15    printf("%4d: ", ++cnt); //1행 처음에 번호 1 출력
16    while ((ch = fgetc(f)) != EOF)
17    {
18       putchar(ch); //putc(ch, stdout);
19       if (ch == '\n') //2행부터 행 처음에 행 번호 출력
20          printf("%4d: ", ++cnt);
21    }
22    printf("\n");
23    fclose(f);
24
25    return 0;
26 }
```

새로운 줄에 이동했으면 다시 처음에 행 번호 ++cnt 출력

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3:
 4: int main(void)
 5: {
 6:    FILE* f;

... 중간생략

24:
25:    return 0;
26: }
27:
```

줄 번호와 함께 파일 내용을 그대로 출력