18장 파일 입출력

한빛미디어 Hanbit Media, Inc.

18-1 파일 개방과 입출력

❖ 파일 개방과 폐쇄

파일을 열고 닫는 프로그램 소스 코드 예제18-1.c

```c
#include <stdio.h>

int main(void)
{
    FILE *fp;                                   // 파일 포인터

    fp = fopen("a.txt", "r");                   // a.txt 파일을 읽기 전용으로 개방
    if (fp == NULL)                             // fp가 널 포인터면 파일 개방 실패
    {
        printf("파일이 열리지 않았습니다.\n");    // 안내 메시지 출력
        return 1;                               // 프로그램 종료
    }
    printf("파일이 열렸습니다.\n");
    fclose(fp);                                 // 파일 닫기

    return 0;
}
```

```c
fp = fopen("a.txt", "r");
```

개방할 파일명 / 개방 모드

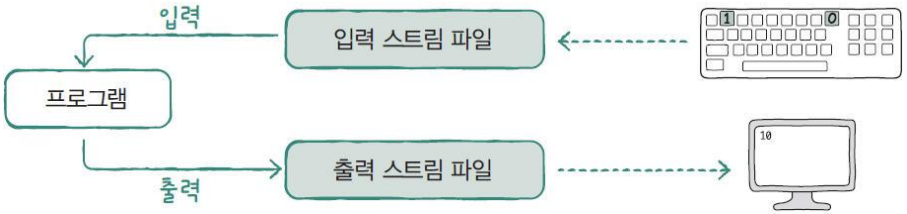
실행결과

```
파일이 열리지 않았습니다.
```

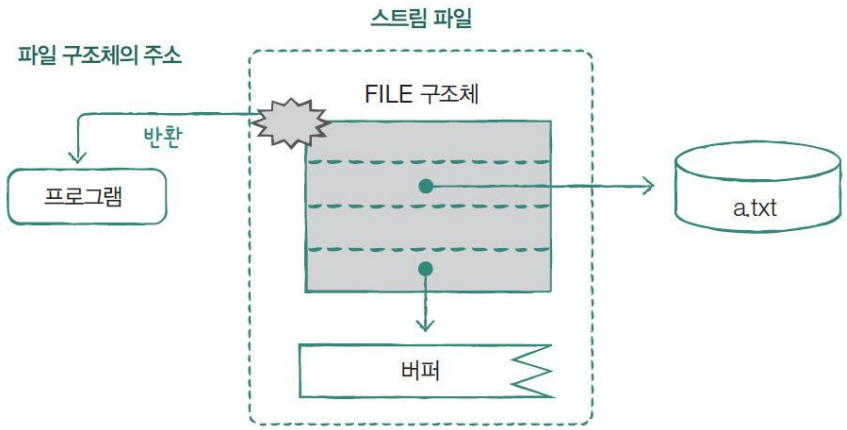
18- 1 파일 개방과 입출력

❖ 스트림 파일과 파일 포인터

- 파일 입출력은 스트림 파일을 통해서 수행한다.



- 파일을 개방하면 스트림 파일을 만들고 파일 포인터를 반환한다.

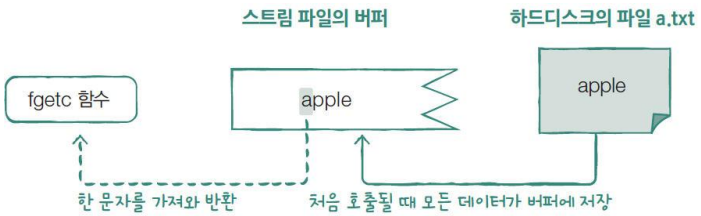


18- 1 파일 개방과 입출력

❖ 문자 입력 함수 fgetc (1/2)

파일의 내용을 화면에 출력하기 소스 코드 예제18-2.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      FILE *fp;
06      int ch;
07
08      fp = fopen("a.txt", "r");
09      if (fp == NULL)
10      {
11          printf("파일이 열리지 않았습니다.\n");
12          return 1;
13      }
14
15      while (1)
16      {
17          ch = fgetc(fp);
18          if (ch == EOF)
19          {
20              break;
21          }
22          putchar(ch);
23      }
24      fclose(fp);
25
26      return 0;
27  }
```

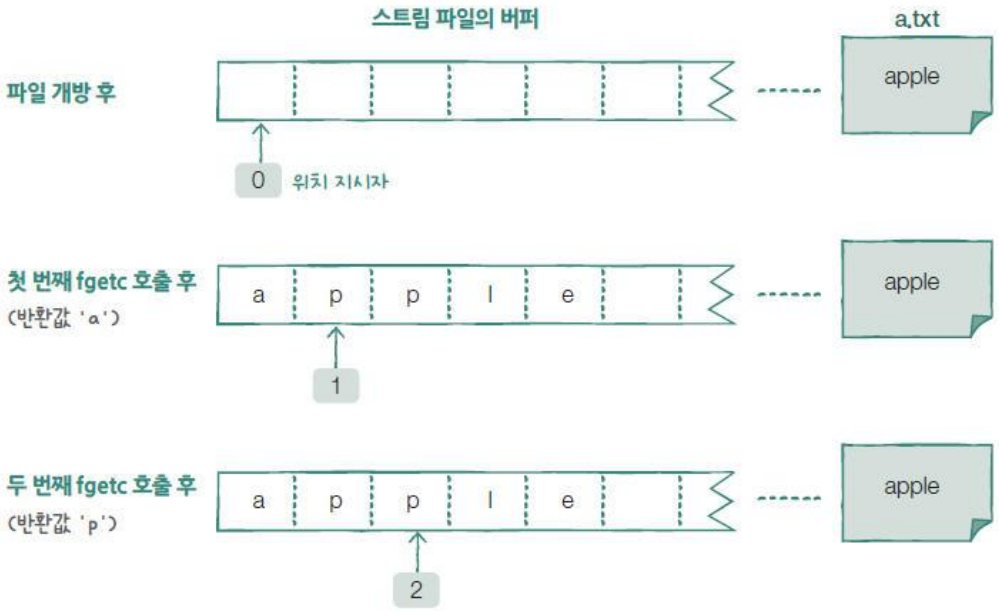


실행결과

```
apple
```

❖ 문자 입력 함수 fgetc (2/2)

- 버퍼의 내용은 위치 지시자가 증가하면서 차례로 가져온다.



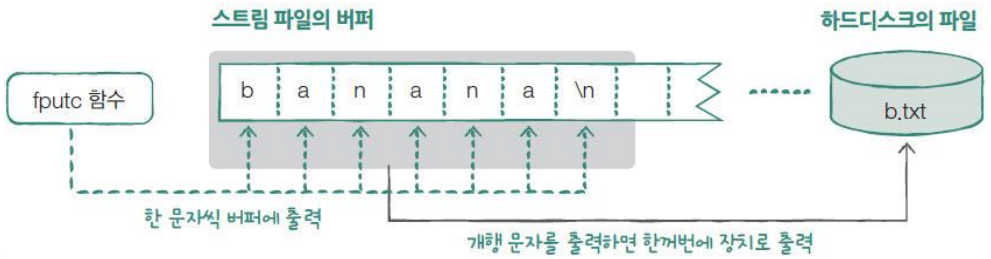
- 버퍼의 내용을 모두 읽으면 EOF(-1)를 반환한다.

18- 1 파일 개방과 입출력

❖ 문자 출력 함수 fputc

문자열을 한 문자씩 파일로 출력하기 소스 코드 예제18-3.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      FILE *fp;
06      char str[] = "banana";
07      int i;
08
09      fp = fopen("b.txt", "w");
10      if (fp == NULL)
11      {
12          printf("파일을 만들지 못했습니다.\n");
13          return 1;
14      }
15
16      i = 0;
17      while (str[i] != '\0')
18      {
19          fputc(str[i], fp);
20          i++;
21      }
22      fputc('\n', fp);
23      fclose(fp);
24
25      return 0;
26  }
```



18-1 파일 개방과 입출력

❖ 기본적으로 개방되는 표준 입출력 스트림 파일 (1/2)

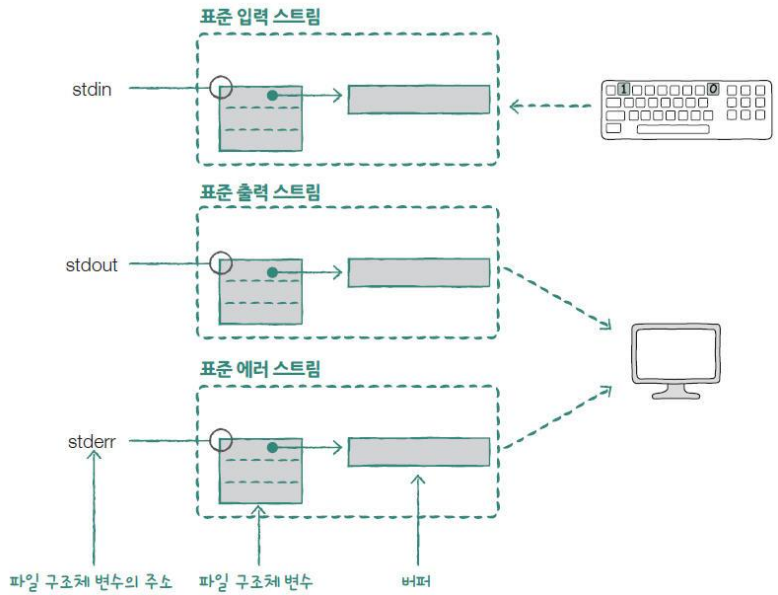
표준 입출력 스트림을 사용한 문자열 입력 소스 코드 예제18-4.c

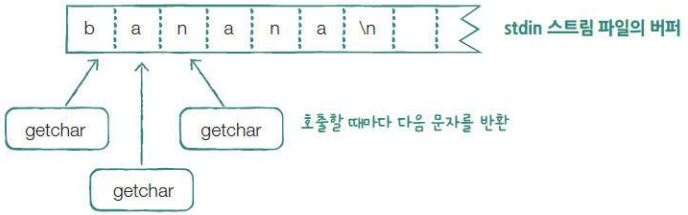
```c
#include <stdio.h>

int main(void)
{
    int ch;

    while (1)
    {
        ch = getchar();
        if (ch == EOF)
        {
            break;
        }
        putchar(ch);
    }

    return 0;
}
```





실행결과

```
banana
banana
^Z
```

18-1 파일 개방과 입출력

❖ 기본적으로 개방되는 표준 입출력 스트림 파일 (2/2)

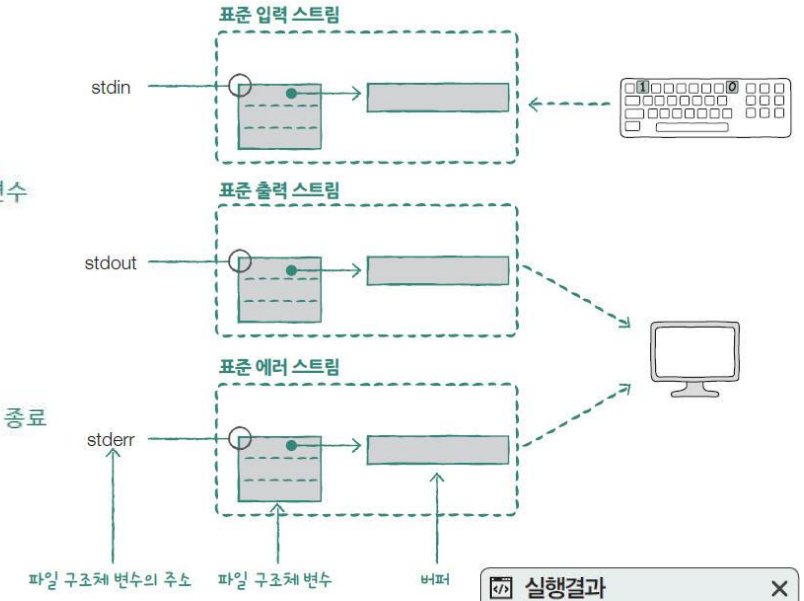
stdin과 stdout을 사용한 문자 입출력 소스 코드 예제18-5.c

```c
#include <stdio.h>

int main(void)
{
    int ch;                     // 입력한 문자를 저장할 변수

    while (1)
    {
        ch = fgetc(stdin);      // 키보드에서 문자 입력
        if (ch == EOF)          // <Ctrl> + <Z>로 입력 종료
        {
            break;
        }
        fputc(ch, stdout);      // 화면에 문자 출력
    }

    return 0;
}
```

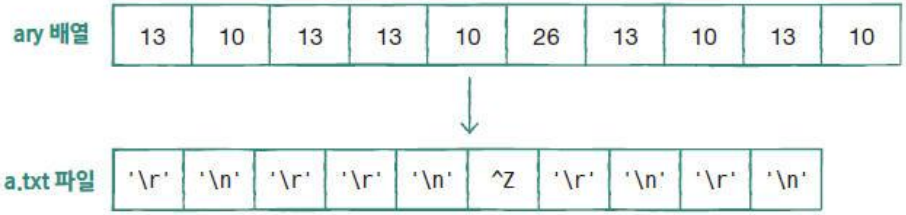


실행결과

```
rabbit
rabbit
turtle
turtle
^Z
```

❖ 텍스트 파일과 바이너리 파일 (1/2)

파일의 형태와 개방 모드가 다른 경우 소스 코드 예제18-6.c

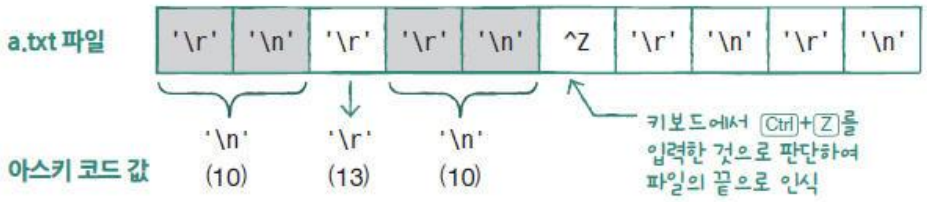



```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      FILE *fp;
06      int ary[10] = { 13, 10, 13, 13, 10, 26, 13, 10, 13, 10 };
07      int i, res;
08
09      fp = fopen("a.txt", "wb");    // 바이너리 파일로 개방
10      for (i = 0; i < 10; i++)
11      {
12          fputc(ary[i], fp);        // 배열 요소의 각 값에 해당하는 아스키 문자 출력
13      }
14      fclose(fp);                   // 파일 닫음
15
```

❖ 텍스트 파일과 바이너리 파일 (2/2)

파일의 형태와 개방 모드가 다른 경우 소스 코드 예제18-6.c

```
16      fp = fopen("a.txt", "rt");
17      while (1)
18      {
19          res = fgetc(fp);
20          if (res == EOF) break;
21          printf("%4d", res);
22      }
23      fclose(fp);
24
25      return 0;
26  }
```



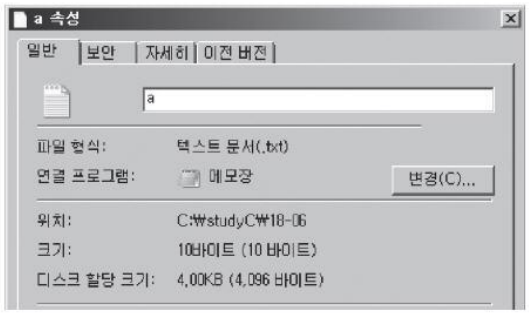
실행결과

텍스트 파일로 개방하여 읽은 데이터

10 13 10

처음 출력한 바이너리 파일의 크기

- 탐색기에서 마우스 오른쪽 버튼으로 속성 확인



18- 1 파일 개방과 입출력

❖ + 개방 모드, fseek, rewind, feof 함수 (1/3)

a+ 모드로 파일의 내용을 확인하며 출력 소스 코드 예제18-7.c

```
01  #include <stdio.h>
02  #include <string.h>
03
04  int main(void)
05  {
06      FILE *fp;
07      char str[20];
08
09      fp = fopen("a.txt", "a+");
10      if (fp == NULL)
11      {
12          printf("파일을 만들지 못했습니다.\n");
13          return 1;
14      }
15
```

개방 모드	파일이 있을 때
r+	텍스트 파일에 읽고 쓰기 위해 개방
w+	텍스트 파일의 내용을 지우고 읽거나 쓰기 위해 개방
a+	텍스트 파일을 읽거나 파일의 끝에 추가하기 위해 개방

처음 실행할 때 실행결과

```
과일 이름 : apple
과일 이름 : banana
과일 이름 : list
apple
banana
과일 이름 : end
```

두 번째 실행할 때 실행결과

```
과일 이름 : strawberry
과일 이름 : list
apple
banana
strawberry
과일 이름 : end
```

❖ + 개방 모드, fseek, rewind, feof 함수 (2/3)

a+ 모드로 파일의 내용을 확인하며 출력 소스 코드 예제18-7.c

```
16      while (1)
17      {
18          printf("과일 이름 : ");
19          scanf("%s", str);
20          if (strcmp(str, "end") == 0)
21          {
22              break;
23          }
24          else if (strcmp(str, "list") == 0)
25          {
26              fseek(fp, 0, SEEK_SET);
27              while (1)
28              {
29                  fgets(str, sizeof(str), fp);
30                  if (feof(fp))
31                  {
32                      break;
33                  }
34                  printf("%s", str);
35              }
36          }
37          else
38          {
39              fprintf(fp, "%s\n", str);
40          }
41      }
42      fclose(fp);
43
44      return 0;
45  }
```

❖ + 개방 모드, fseek, rewind, feof 함수 (3/3)

- fseek 함수는 + 모드에서 읽고 쓰기를 바꿀 때 필요하다.

fseek 함수 원형

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

whence에 사용할 수 있는 값과 의미

값	매크로명	기준 위치	오프셋값
0	SEEK_SET	파일의 처음	양수만 가능
1	SEEK_CUR	파일의 현재 위치	양수와 음수 모두 가능
2	SEEK_END	파일의 끝	음수만 가능

```
fseek(fp, 0, SEEK_SET);
```

→

```
rewind(fp);
```

- feof 함수는 파일의 끝이면 참(0이 아닌 값)을 반환한다.

마무리

키워드로 끝내는 핵심 포인트

- **fopen** 함수가 파일을 개방하면 메모리에 스트림 파일을 만든다.
- **스트림 파일**은 프로그램과 장치를 연결하며 버퍼에 데이터를 저장한다.
- **파일 입출력 함수**는 스트림 파일을 통해 입출력을 수행한다.
- **fclose** 함수는 개방한 스트림 파일을 메모리에서 제거한다.

마무리

표로 정리하는 핵심 포인트

표 18-1 파일 개방 함수와 문자 입출력 함수

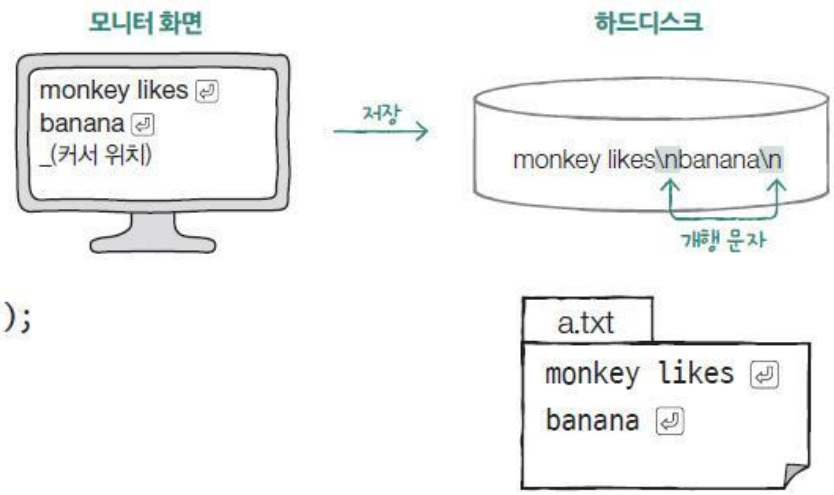
구분	기능	사용 예
파일 열기 fopen	원형	`FILE *fopen(const char *, const char *);`
	사용 예	`FILE *fp;` `fp = fopen("a.txt", "r");`
	반환값	개방에 성공하면 FILE 포인터, 실패하면 널 포인터(NULL)
파일 닫기 fclose	원형	`int fclose(FILE *);`
	사용 예	`fclose(fp);`
	반환값	성공하면 0, 오류가 발생한 경우 EOF
문자 입력 fgetc	원형	`int fgetc(FILE *);`
	사용 예	`int ch;` `ch = fgetc(fp);`
	반환값	입력한 문자, 오류나 파일에 데이터가 없을 때 EOF
문자 출력 fputc	원형	`int fputc(int, FILE *);`
	사용 예	`fputc(ch, ofp);`
	반환값	출력한 문자, 오류가 발생한 경우 EOF

18- 2 다양한 파일 입출력 함수

❖ fgets와 fputs : 한 줄씩 입출력 (1/2)

여러 줄의 문장을 입력하여 한 줄로 출력 소스 코드 예제18-8.c

```
01  #include <stdio.h>
02  #include <string.h>
03
04  int main(void)
05  {
06      FILE *ifp, *ofp;
07      char str[80];
08      char *res;
09
10      ifp = fopen("a.txt", "r");
11      if (ifp == NULL)
12      {
13          printf("입력 파일을 열지 못했습니다.\n");
14          return 1;
15      }
16
17      ofp = fopen("b.txt", "w");
18      if (ofp == NULL)
19      {
20          printf("출력 파일을 열지 못했습니다.\n");
21          return 1;
22      }
23
```



18- 2 다양한 파일 입출력 함수

❖ fgets와 fputs : 한 줄씩 입출력 (2/2)

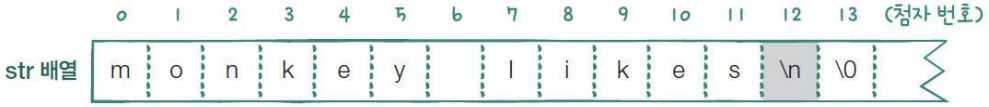
여러 줄의 문장을 입력하여 한 줄로 출력 소스 코드 예제18-8.c

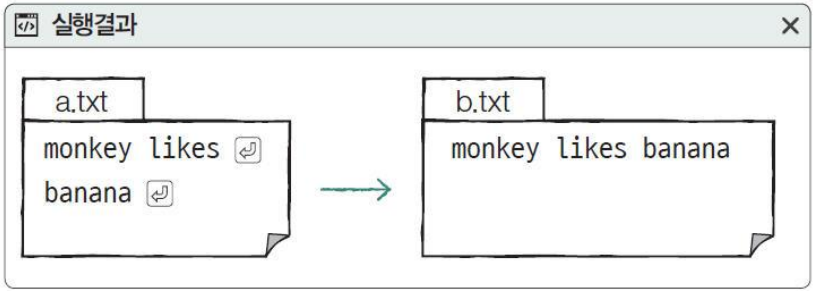
```
24      while (1)
25      {
26          res = fgets(str, sizeof(str), ifp);
27          if (res == NULL)
28          {
29              break;
30          }
31          str[strlen(str) - 1] = '\0';
32          fputs(str, ofp);
33          fputs(" ", ofp);
34      }
35
36      fclose(ifp);
37      fclose(ofp);
38
39      return 0;
40  }
```

```
str[strlen(str) - 1] = '\0';
```

↓

개행 문자가 저장된 위치의 첨자



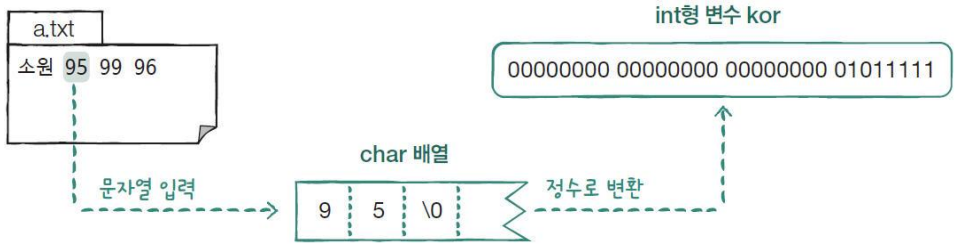


18- 2 다양한 파일 입출력 함수

❖ fscanf, fprintf : 다양한 형태의 입출력 (1/2)

다양한 자료형을 형식에 맞게 입출력 소스 코드 예제18-9.c

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      FILE *ifp, *ofp;
06      char name[20];
07      int kor, eng, math;
08      int total;
09      double avg;
10      int res;
11
12      ifp = fopen("a.txt", "r");
13      if (ifp == NULL)
14      {
15          printf("입력 파일을 열지 못했습니다.\n");
16          return 1;
17      }
18
19      ofp = fopen("b.txt", "w");
20      if (ofp == NULL)
21      {
22          printf("출력 파일을 열지 못했습니다.\n");
23          return 1;
24      }
25
```



a.txt

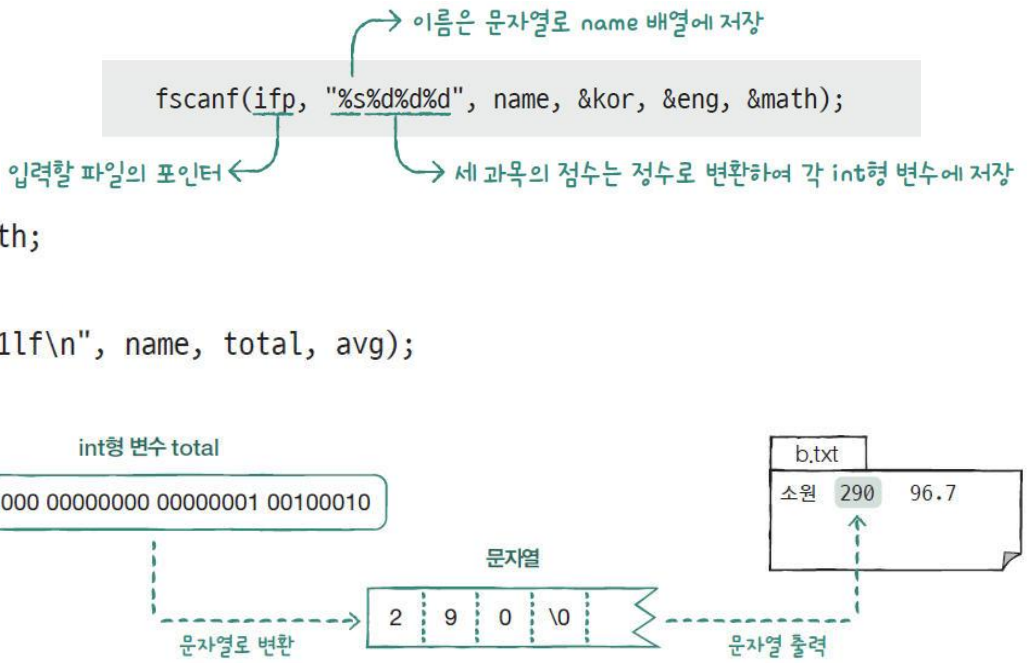
```
소원 95 99 96
유주 80 85 94
신비 92 76 93
```

18- 2 다양한 파일 입출력 함수

❖ fscanf, fprintf : 다양한 형태의 입출력 (2/2)

다양한 자료형을 형식에 맞게 입출력 소스 코드 예제18-9.c

```
26      while (1)
27      {
28          res = fscanf(ifp, "%s%d%d%d", name, &kor, &eng, &math);
29          if (res == EOF)
30          {
31              break;
32          }
33          total = kor + eng + math;
34          avg = total / 3.0;
35          fprintf(ofp, "%s%5d%7.1lf\n", name, total, avg);
36      }
37
38      fclose(ifp);
39      fclose(ofp);
40
41      return 0;
42  }
```

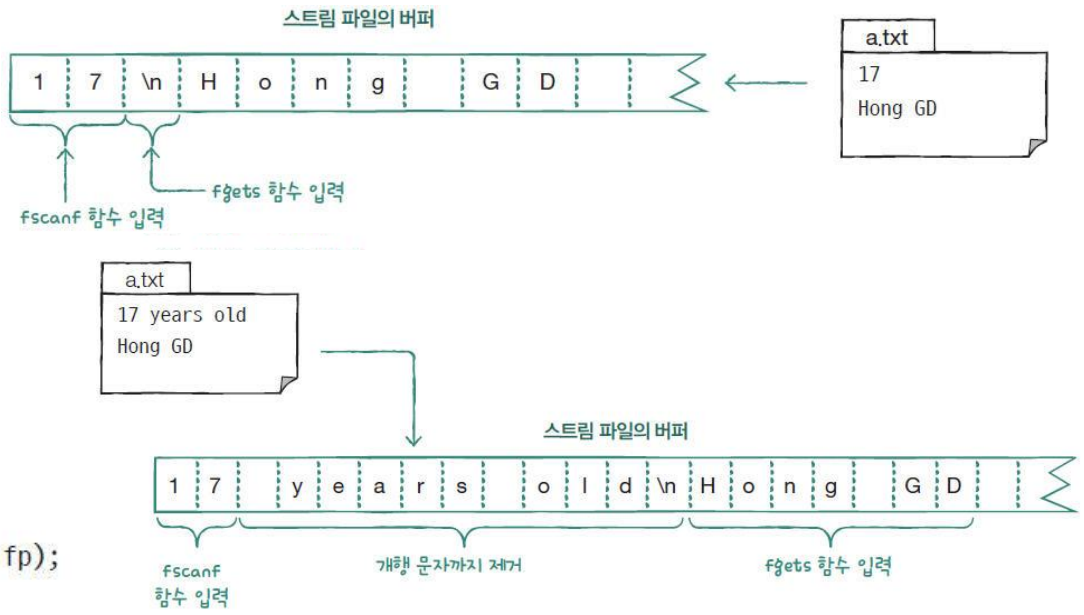


18- 2 다양한 파일 입출력 함수

❖ 스트림 파일의 버퍼 공유 문제와 fflush 함수

버퍼를 공유하므로 발생하는 문제 소스 코드 예제18-10.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     FILE *fp;
06     int age;
07     char name[20];
08
09     fp = fopen("a.txt", "r");
10
11     fscanf(fp, "%d", &age);
12     fgets(name, sizeof(name), fp);
13
14     printf("나이 : %d, 이름 : %s", age, name);
15     fclose(fp);
16
17     return 0;
18 }
```




```
while (fgetc(fp) != '\n') { }
```

이 코드를 11-12행 사이에 추가

실행결과

```
나이 : 17, 이름 :
_
```

18- 2 다양한 파일 입출력 함수

❖ fread와 fwrite 함수 (1/2)

fprintf와 fwrite 함수의 차이 소스 코드 예제18-11.c

```
01 #include <stdio.h>
02
03 int main(void)
04 {
05     FILE *afp, *bfp;
06     int num = 10;
07     int res;
08
09     afp = fopen("a.txt", "wt");
10     fprintf(afp, "%d", num);
11
12     bfp = fopen("b.txt", "wb");
13     fwrite(&num, sizeof(num), 1, bfp);
14
15     fclose(afp);
16     fclose(bfp);
17
18     bfp = fopen("b.txt", "rb");
19     fread(&res, sizeof(res), 1, bfp);
20     printf("%d", res);
21
22     fclose(bfp);
23
24     return 0;
25 }
```

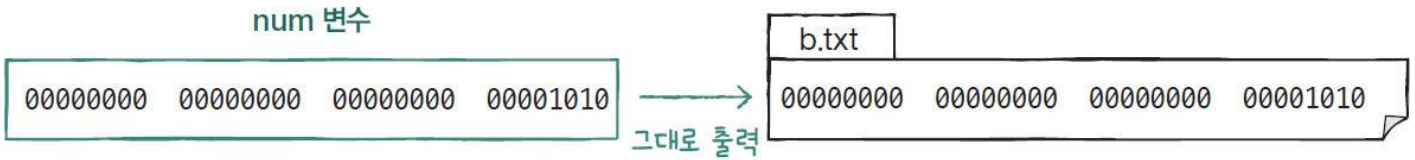
실행결과

```
10
```

❖ fread와 fwrite 함수 (2/2)

- fread와 fwrite 는 메모리의 데이터를 변환없이 입출력한다.
  - fprintf 함수가 변환하여 파일에 출력하는 경우

  - fwrite가 변환 없이 파일에 출력하는 경우



마무리

키워드로 끝내는 핵심 포인트

- **fgets** 함수는 한 줄씩 입력하며 데이터를 모두 읽으면 NULL을 반환한다.
- **fscanf** 함수와 **fprintf** 함수는 scanf, printf 함수와 사용법이 비슷하다 .
- **fflush** 함수는 출력할 때 사용하며 스트림 버퍼의 내용을 즉시 장치로 기록한다.
- **fread**와 **fwrite** 함수는 데이터의 크기를 지정해 입출력할 수 있다.

마무리

표로 정리하는 핵심 포인트

표 18-2 다양한 파일 입출력 함수

구분	기능	사용 예
한 줄 입력 fgets	원형	`char *fgets(char *, int, FILE *);`
	사용 예	`char str[20];` `fgets(str, sizeof(str), fp);`
	반환값	입력한 char 배열, 파일의 끝이면 NULL
문자열 출력 fputs	원형	`int fputs(const char *, FILE *);`
	사용 예	`fputs(str, ofp);`
	반환값	출력에 성공하면 음수가 아닌 값, 실패하면 EOF
변환 입력 fscanf	원형	`int fscanf(FILE *, const char *, ...);`
	사용 예	`int age;` `fscanf(fp, "%d", &age);`
	반환값	입력에 성공한 데이터 수, 파일에 데이터가 없을 때 EOF
변환 출력 fprintf	원형	`int fprintf(FILE *, const char *, ...);`
	사용 예	`fprintf(ofp, "나이 : %d\n", age);`
	반환값	출력한 문자의 바이트 수, 실패하면 음수