C Language

파일 입출력

Doo-ok Seo

clickseo@kw.ac.kr

http://www.clickseo.com

목 차

- 파일 입출력 개념
- 파일 입출력 함수
- 기타 파일 처리 함수

파일 입출력 개념

- 파일 입출력 개념
  - 파일의 기본 개념
  - 파일 시스템의 개요
  - FILE 구조체
  - 파일 테이블
  - 파일 열기 및 닫기 : fopen(), fclose() 함수
- 파일 입출력 함수
- 기타 파일 처리 함수

파일 입출력 개념

파일의 기본 개념

- 하나의 단위로 취급해야 하는 데이터들의 외부적 컬렉션이다.

- 파일의 종류
  - **텍스트 파일**
    - 모든 데이터가 그래픽 문자로 저장
    - 라인 단위로 구성되어 있으며, 각 라인은 개행 문자('\n')로 끝난다.
  - **이진 파일**
    - 데이터가 정수 혹은 부동소수점 숫자와 같은 컴퓨터 내부 표현이 그대로 저장된다.

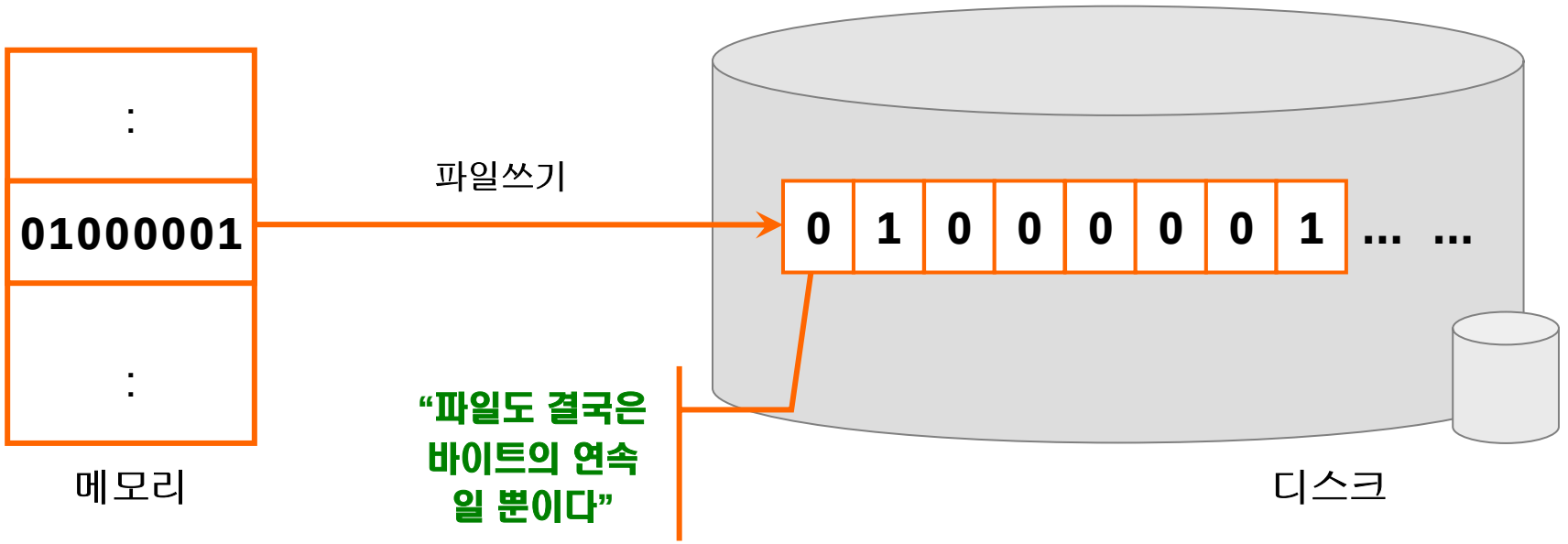
- 파일은 보조 혹은 이차 저장장치에 저장된다.

파일 입출력 개념 (cont'd)

파일의 기본 개념 (cont'd)

메모리와 파일의 관계

- 파일은 디스크에 어떤 모습으로 존재하는가?

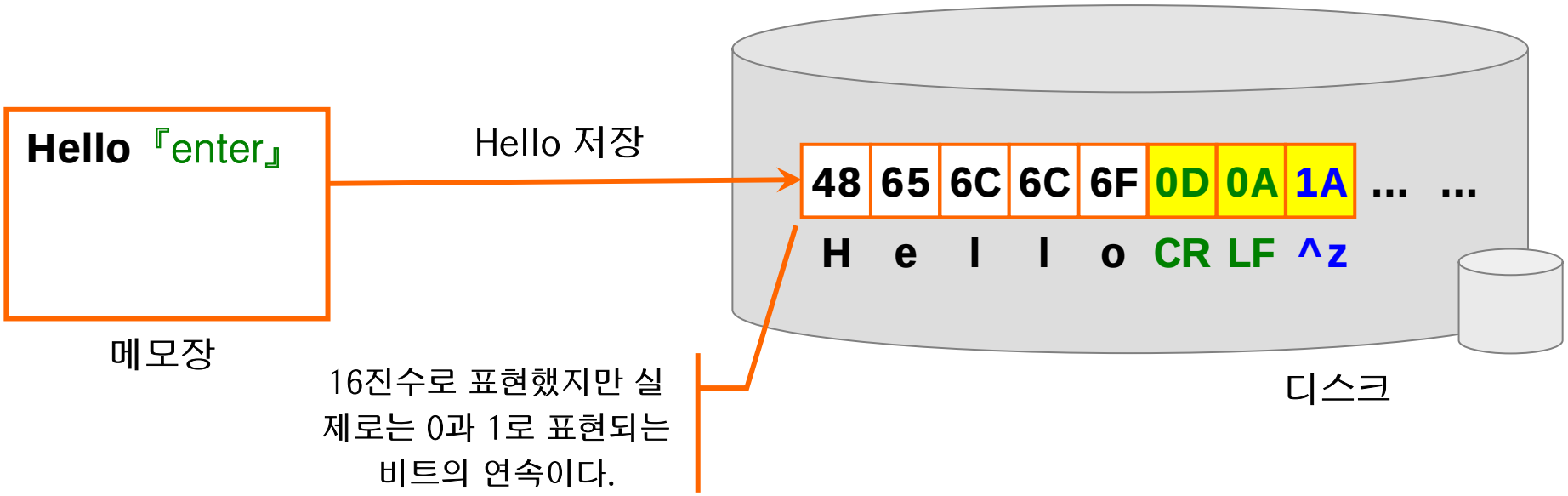


파일 입출력 개념 (cont'd)

● 파일의 기본 개념 (cont'd)

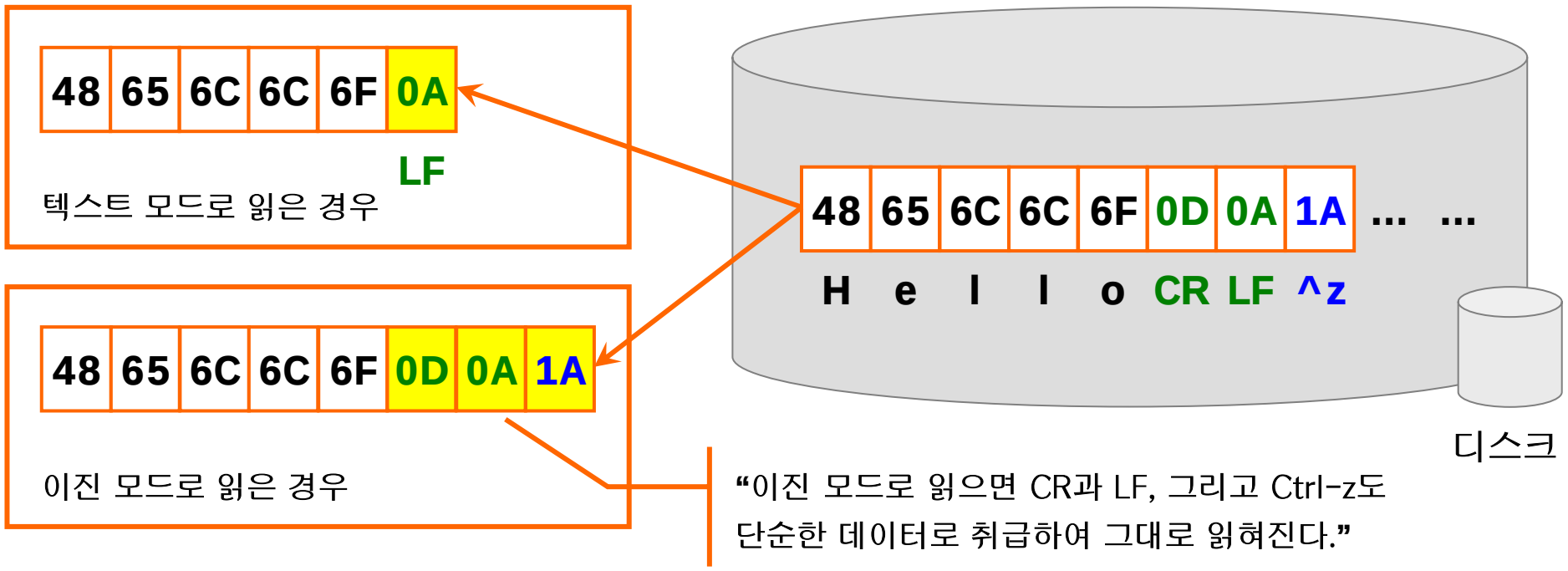
❍ 텍스트 파일의 저장

- **"사용자가 입력한 문자는 ASCII 코드에 대응되는 이진수로 저장된다."**
  - 그림에서는 편의상 16진수로 표현했을 뿐 실제로는 이진수의 나열이다.



파일 입출력 개념 (cont'd)

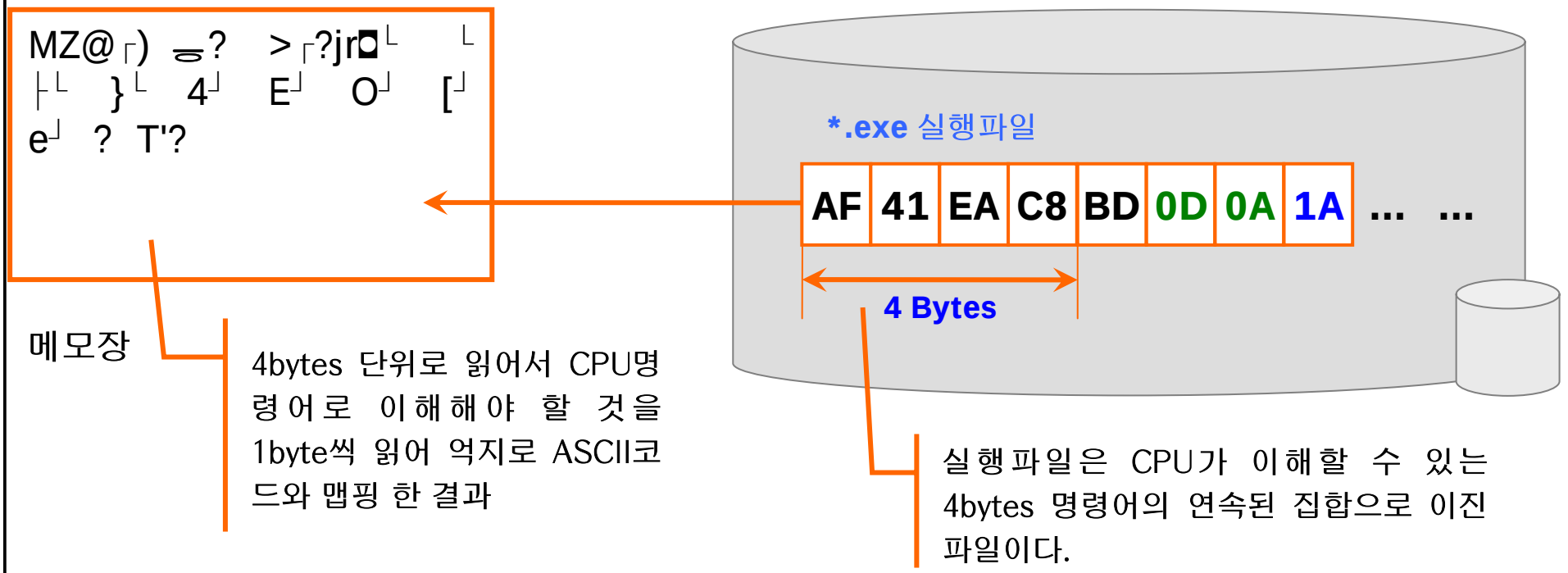
- **파일의 기본 개념 (cont'd)**
  - 텍스트 모드와 이진 모드의 차이



파일 입출력 개념 (cont'd)

● 파일의 기본 개념 (cont'd)

❍ 텍스트 모드와 이진 모드의 차이 (cont'd)



파일 입출력 개념 (cont'd)

- **파일 시스템의 개요**
  - 파일 스트림
    - 디스크에 존재하는 물리적인 파일도 하나의 스트림으로 다룬다.
    - 파일 스트림은 열고 나서 작업을 수행하고 난 다음 반드시 닫아야 한다.
  - **고수준 파일 입출력 함수**
    - 버퍼형 파일 입출력 함수 : buffered file I/O
      - **"데이터를 파일에서 일정 블록 단위로 한 번에 읽어 메모리 상에 확보된 버퍼 영역에 위치시키는 것"**
      - 버퍼 내부의 데이터를 다양한 방법으로 조작하고 활용 가능
    - 호환성이 높은 프로그램을 훨씬 유연하게 제작할 수 있다.
  - **저수준 파일 입출력 함수**
    - 비 버퍼형 파일 입출력 함수
      - **"입출력의 요구가 있을 때마다 bytes 단위로 읽기/쓰기를 행하는 것"**
    - 입 출력 속도가 빠르다.
    - 하드웨어에 관한 부가적인 지식을 어느 정도 요구하고 개발이 어려운 편이다.

파일 입출력 개념 (cont'd)

- **FILE 구조체**
  - stdio.h에 정의되어 있는 FILE 구조체

```
typedef struct  {
  int             level;   /* fill/empty level of buffer */
  unsigned        flags;   /* File status flags          */
  char            fd;      /* File descriptor            */
  unsigned char   hold;    /* Ungetc char if no buffer   */
  int             bsize;   /* Buffer size                */
  unsigned char   *buffer; /* Data transfer buffer       */
  unsigned char   *curp;   /* Current active pointer     */
  unsigned        istemp;  /* Temporary file indicator   */
  short           token;   /* Used for validity checking */
} FILE;                    /* This is the FILE object    */
```

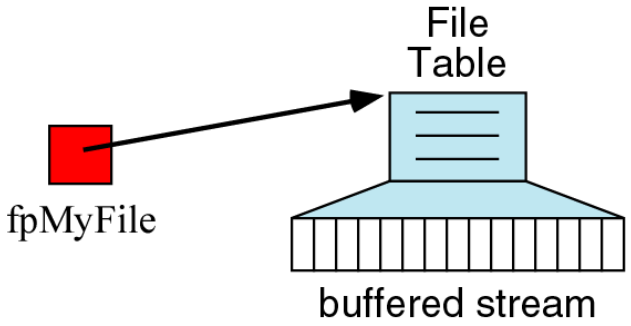
- buffer : 버퍼의 메모리 주소를 가리키는 포인터 변수
- curp : 버퍼 내의 위치를 가리키는 포인터 변수

파일 입출력 개념 (cont'd)

- **파일 테이블**
  - 프로그램에서 파일을 사용하는 부분과 외부 파일 사이의 연결
  - 파일은 표준 **FILE** 형으로 정의
    - 표준 입출력 헤더 파일에 정의되어 있다 → **<stdio.h>**

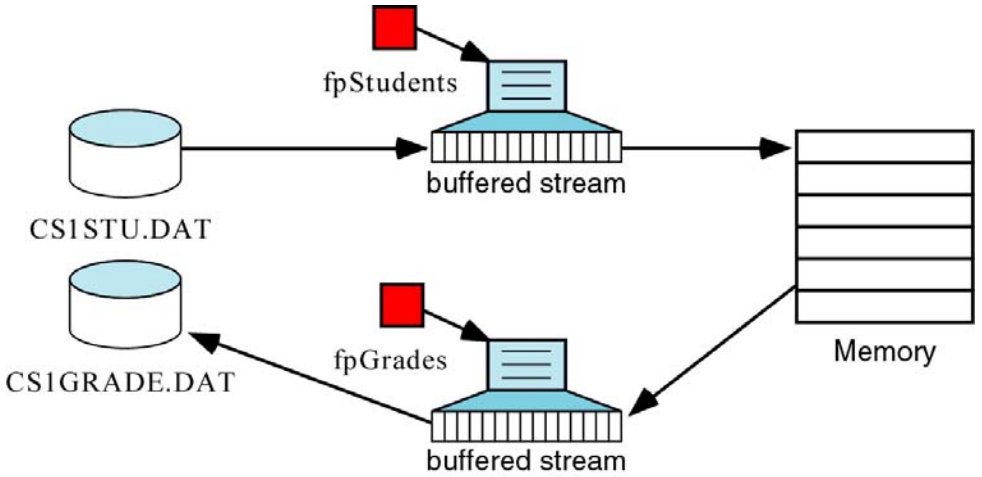
```
FILE      *filename;
```

      - 식별자가 모두 대문자이며, 이는 미리 정의된 형의 한 가지를 나타낸다.
      - 별표(*)가 형의 정의에 나타나기 때문에 이 형은 주소를 나타낸다.
    - **파일 테이블은 파일 이름, 파일 버퍼의 위치, 파일의 현재 상태 등의 정보를 가지고 있다.**



파일 입출력 개념 (cont'd)

- **사용자 파일들**
  - **FILE** 형을 사용하여 필요한 임의의 외부 파일을 정의



"파일은 오직 하나의 스트림과 연결된다.

마찬가지로 한 스트림도 오직 하나의 외부 파일과 연결된다."

파일 입출력 개념 (cont'd)

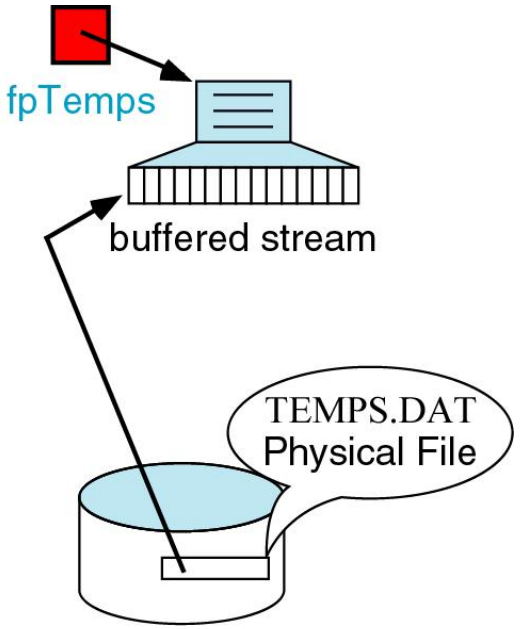
● 파일 열기 : fopen()

- 외부 파일과 프로그램 사이에 연결을 만든다.
- 파일 테이블을 생성하여 파일 처리에 필요한 정보를 저장한다.

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

- filename : 대상 파일의 경로를 포함한 이름
- mode : 파일의 열기 모드
  - "대상 파일이 없거나 오류가 발생한다면 NULL을 반환한다."

```
FILE *fpTemps;

if (( fpTemps = fopen("TEMPS.DAT", "w")) == NULL
{
   printf("File Open Error₩n");
   exit(1);
}
```



파일 입출력 개념 (cont'd)

파일 열기 (cont'd)

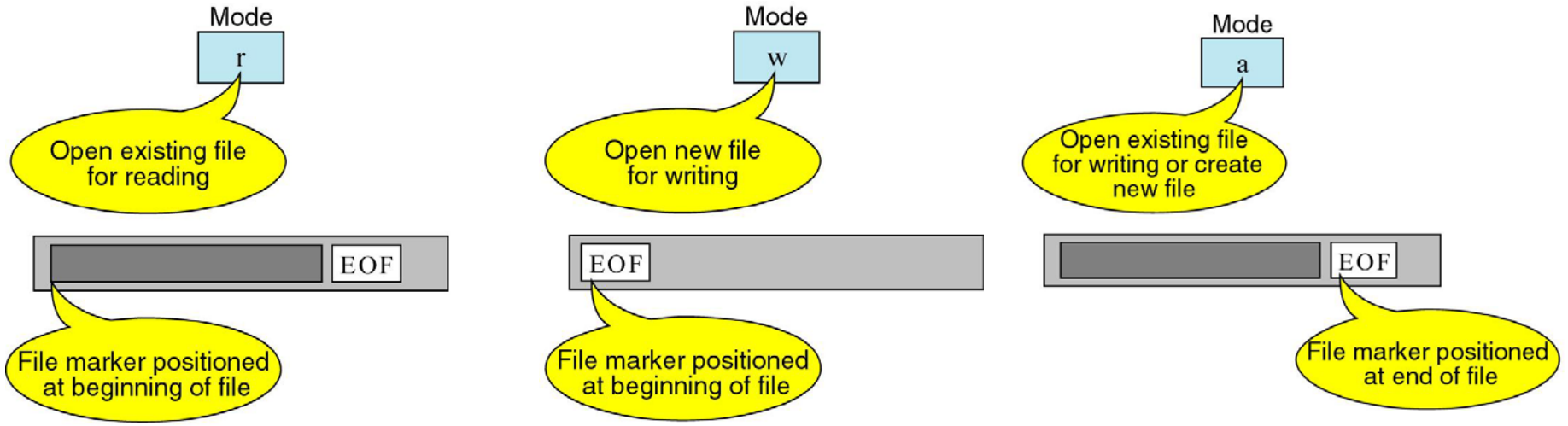
- fopen() 함수의 파일 열기 모드 : **텍스트 모드 (text mode)**

모 드	의 미	비 고
r	**읽기 전용 (Read Only)** 파일이 없으면 NULL을 반환	쓰기 불가
w	**쓰기 전용 (Write Only)** 파일이 존재하지 않으면 새 파일을 생성 기존 파일이 있으면 그 내용은 무시하고 처음부터 새로 쓴다.	읽기 불가
a	**추가 모드 (Append Only)** 파일이 존재하지 않으면 새 파일을 생성 기존 파일이 있으면 기존 파일의 끝에 추가만 가능하다.	읽기 불가
r+	기존 파일에 대한 읽기와 쓰기가 모두 가능하도록 파일 개방 파일이 없으면 NULL을 반환	읽기 + 쓰기
w+	무조건 새로운 파일을 생성하여 읽기와 쓰기가 가능하도록 파일을 연다.	읽기 + 쓰기
a+	기존 파일의 끝에서부터 읽기와 쓰기가 가능하도록 파일을 열고, 파일이 없으면 새로 생성한다.	**이전 부분은 쓰기 불가**

파일 입출력 개념 (cont'd)

- **파일 열기 (cont'd)**
  - fopen() 함수의 파일 열기 모드 : **이진 모드 (binary mode)**

모 드	의 미
rb, wb, ab	이진 모드로 파일을 개방하고 텍스트 모드에서의 r, w, a와 같은 의미
r+b, rb+	이진 모드로 파일을 개방하고 텍스트 모드에서의 r+, w+, a+와 동일한 의미
w+b, wb+	
a+b, ab+	



파일 입출력 개념 (cont'd)

- **파일 닫기 : fclose()**
  - 파일을 더 이상 사용할 필요가 없을 경우에는 파일을 닫아서 버퍼 공간과 같은 시스템 자원을 반납하여야 한다.

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

호출 성공 : 0 을 반환
호출 실패 : **EOF** 를 반환

```
#include <stdio.h>

int  main (void)
{
        FILE  *fp;
        ...
        fp  =  fopen("test.dat",  "w");
        ...
        fclose(fp);
        ...
}
```

파일 입출력 개념 (cont'd)

- 전형적인 파일 처리 작업 형태

```
#include <stdio.h>

int main()
{
        FILE  *fin,  *fout;
        if (fin  =  fopen("source.dat",  "r"))  ==  NULL)     {
                ...          // 에러 처리 코드
        }
        if (fout  =  fopen("test.dat",  "w"))  ==  NULL)     {
                ...
        }

        ...          // 대상 파일에 대한 읽고 쓰는 작업을 수행

        fclose(fin);
        fclose(fout);

        return 0;
}
```

파일 입출력 함수

- 파일 입출력 개념
- **파일 입출력 함수**
  - **문자 입출력 함수 : fgetc(), fputc()**
  - **문자열 입출력 함수 : fgets(), fputs()**
  - **서식화된 파일 입출력 함수 : fscanf(), fprintf()**
  - **이진 파일 입출력 함수 : fread(), fwrite()**
  - **랜덤 액세스 함수 : fseek(), ftell()**
- 기타 파일 처리 함수

파일 입출력 함수

● 문자 입출력 함수 : fgetc()

❍ 파일에서 한 문자(1 byte)씩 읽는 함수

```
#include <stdio.h>

int  fgetc (FILE  *fp);
```

호출 성공 : 읽어 들인 한 문자의 byte를 int 형으로 반환
호출 실패 또는 파일의 끝을 만나면 : **EOF** 를 반환

– fp : 대상이 되는 파일 포인터

• 파일 포인터 fp가 가리키는 대상 파일에서 한 문자를 읽어 int 형으로 반환한다.

파일 입출력 함수 (cont'd)

● 문자 입출력 함수 : fputc()

❍ 파일에 한 문자(1byte)씩 쓰는 함수

```
#include <stdio.h>

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

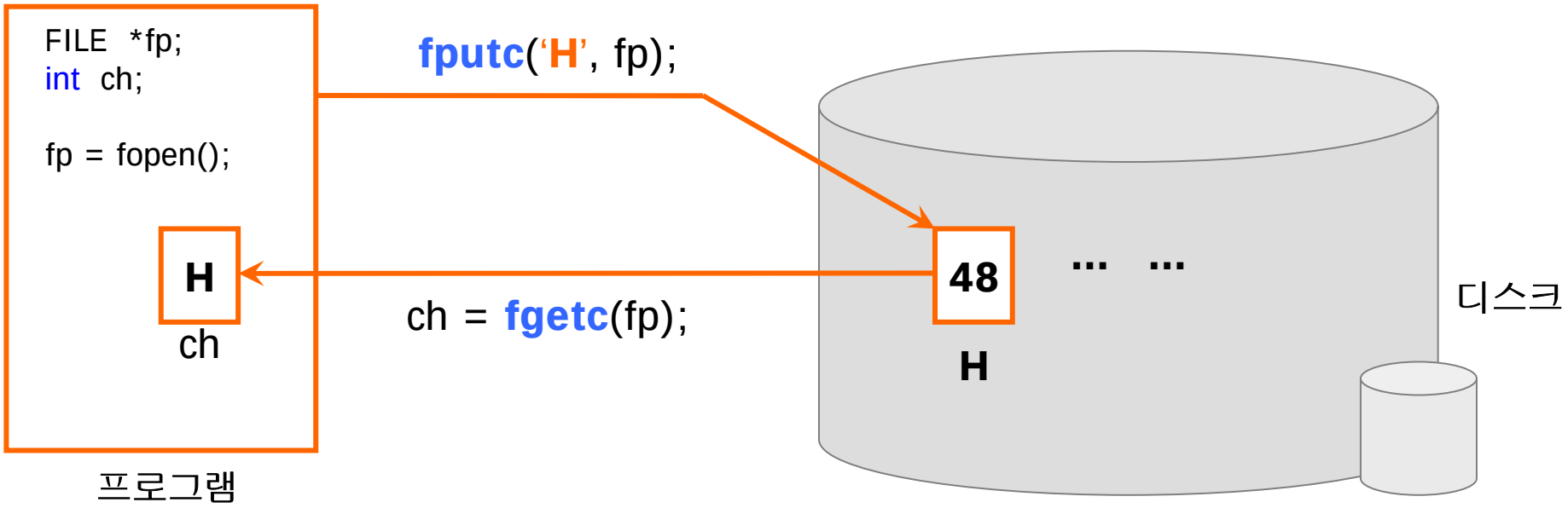
호출 성공 : 기록한 문자를 int형으로 반환
호출 실패 : **EOF** 를 반환

- ch : 파일에 기록할 문자 상수
- fp : 대상이 되는 파일 포인터

- 파일 포인터 fp가 가리키는 대상 파일에 실 인자 ch로 전달된 한 문자를 쓴다.

파일 입출력 함수 (cont'd)

- **문자 입출력 함수 : fgetc()와 fputc()의 동작 과정**



파일 입출력 함수 (cont'd)

- **문자 입출력 함수 : ungetc()**
  - 지정된 문자를 입력 스트림에 되돌려 놓는다(getc와 반대 동작).
  - 이 스트림을 다음에 읽으면 지금 되돌려진 문자가 반환될 것이다.

```
#include <stdio.h>

int  ungetc (int  oneChar,  FILE  *stream);
```

```
option  =  getc(stdin);
if (isdigit(option))
     ungetc(option, stdin);
else
     { ... }
```

파일 입출력 함수 (cont'd)

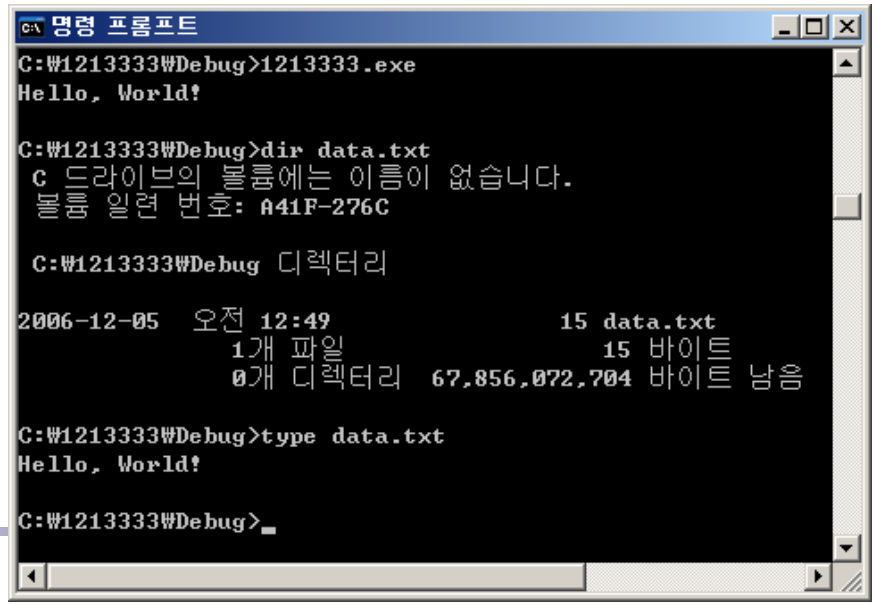
프로그램 예제 : fputc()와 fgetc()에 의한 파일쓰기와 읽기 예 (1/2)

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

int main()
{
        char    ch, *p;
        char    str[] = "Hello, World!\n";
        FILE    *fin, *fout;

        fout = fopen("data.txt", "w");
        if (!fout)
        {
                printf("File Open Error\n");
                exit(1);
        }

        p = str;

        while (*p)
                fputc(*p++, fout);

        fclose(fout);
```

파일 입출력 함수 (cont'd)

프로그램 예제 : fputc()와 fgetc()에 의한 파일쓰기와 읽기 예 (2/2)

```
        fin = fopen("data.txt", "r");
        if (!fin)
        {
                printf("File Open Error\n");
                exit(1);
        }

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

        fclose(fin);

        return 0;
}
```



파일 입출력 함수 (cont'd)

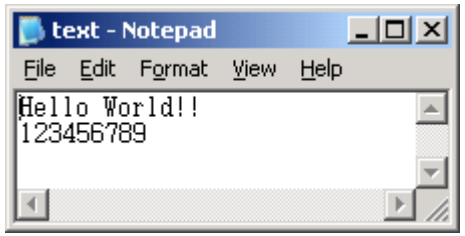
프로그램 예제 : 텍스트 파일 작성하기 (1/2)

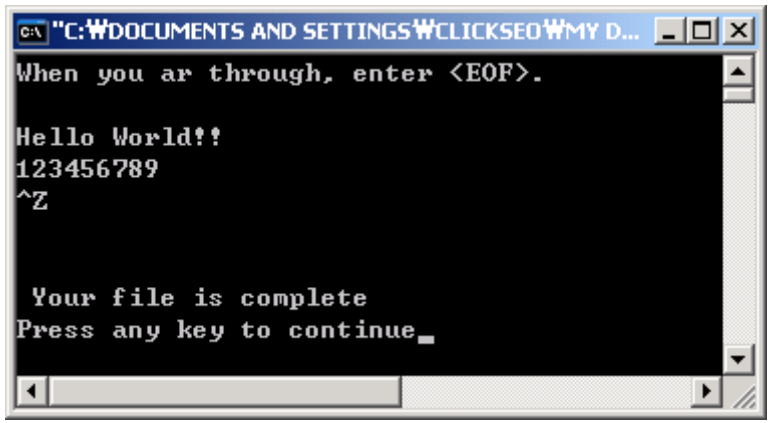
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        FILE *fp;
        int c;
        int closeStatus;

        printf("This program copies your input to a file. \n");
        printf("When you ar through, enter <EOF>. \n\n");

        if(!(fp = fopen("text.txt","w")))
        {
                printf("Error opening FILE1.DAT for writing");
                return(1);
        }

        while((c = getchar()) != EOF)
                fputc(c, fp);
```

파일 입출력 함수 (cont'd)

프로그램 예제 : 텍스트 파일 작성하기 (2/2)

```
        closeStatus = fclose(fp);
        if(closeStatus == EOF)
        {
                printf("Error closing file \a\n");
                return 100;
        }

        printf("\n\n Your file is complete \n");

        return 0;
}
```





파일 입출력 함수 (cont'd)

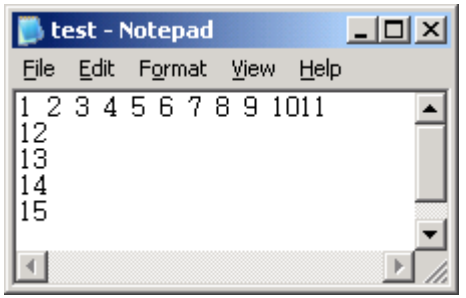
프로그램 예제 : 문자 수와 라인 수 계산하기 (1/2)

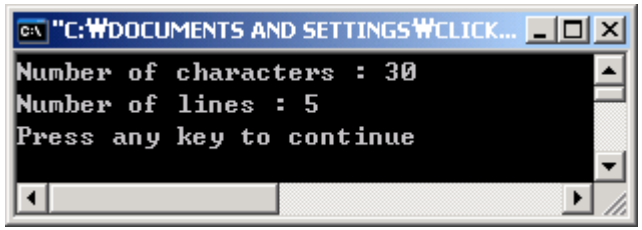
```
#include <stdio.h>

int main(void)
{
        int curCh;
        int preCh;
        int countLn = 0;
        int countCh = 0;
        FILE *fp1;

        if(!(fp1 = fopen("test.txt", "r")))
        {
                printf("Error opening TEXT.TXT for reading");
                return(1);
        }
```

파일 입출력 함수 (cont'd)

프로그램 예제 : 문자 수와 라인 수 계산하기 (2/2)

```
        while((curCh = fgetc(fp1)) != EOF)
        {
                if (curCh != '\n')
                        countCh++;
                else
                        countLn++;
                preCh = curCh;
        }

        if (preCh != '\n')
                countLn++;

        printf("Number of characters : %d \n", countCh);
        printf("Number of lines : %d \n", countLn);

        return 0;
}
```





파일 입출력 함수 (cont'd)

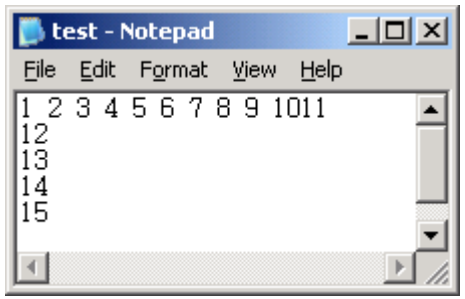
프로그램 예제 : 단어 수 계산하기 (1/2)

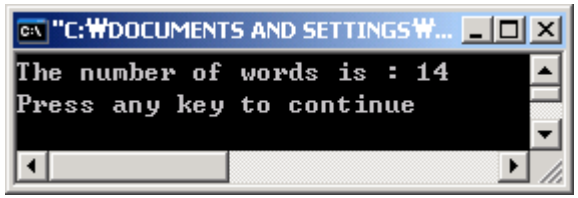
```
#include <stdio.h>

#define WHT_SPC (cur == ' ' || cur == '₩n' || cur == '₩t')

int main(void)
{
        int cur;
        int countWd = 0;
        char word = 'O';
        FILE *fp;

        if(!(fp = fopen("test.txt", "r")))
        {
                printf("Error opening test.txt for reading");
                return(1);
        }
```

파일 입출력 함수 (cont'd)

프로그램 예제 : 단어 수 계산하기 (2/2)

```
        while((cur = fgetc(fp)) != EOF )
        {
                if (WHT_SPC)
                        word = 'O';
                else
                        if (word == 'O')
                        {
                                countWd++;
                                word = 'I';
                        }
        }

        printf("The number of words is : %d \n", countWd);

        fclose(fp);

        return 0;
}
```





파일 입출력 함수 (cont'd)

● 문자열 입출력 함수 : fgets()

❍ 파일에서 문자열 단위로 입력을 수행하는 함수

```
#include <stdio.h>

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

호출 성공 : str의 주소를 반환
호출 실패 : **NULL** 을 반환

- str : 파일에서 읽어 들인 문자열을 저장할 공간에 대한 포인터
- n : 읽어 들일 문자열의 최대 길이
- fp : 대상이 되는 파일 포인터

- fp가 가리키는 파일을 대상으로 n개의 문자를 str이 가리키는 메모리 공간에 읽어 들인다.
- **"줄 바꿈 문자 '\n'을 만나거나 파일의 끝에 도달하면 수행을 멈춘다."**

파일 입출력 함수 (cont'd)

- **문자열 입출력 함수 : fputs()**
  - 파일에서 문자열 단위로 출력을 수행하는 함수

```
#include <stdio.h>

char *fputs(char *str, FILE *fp);
```

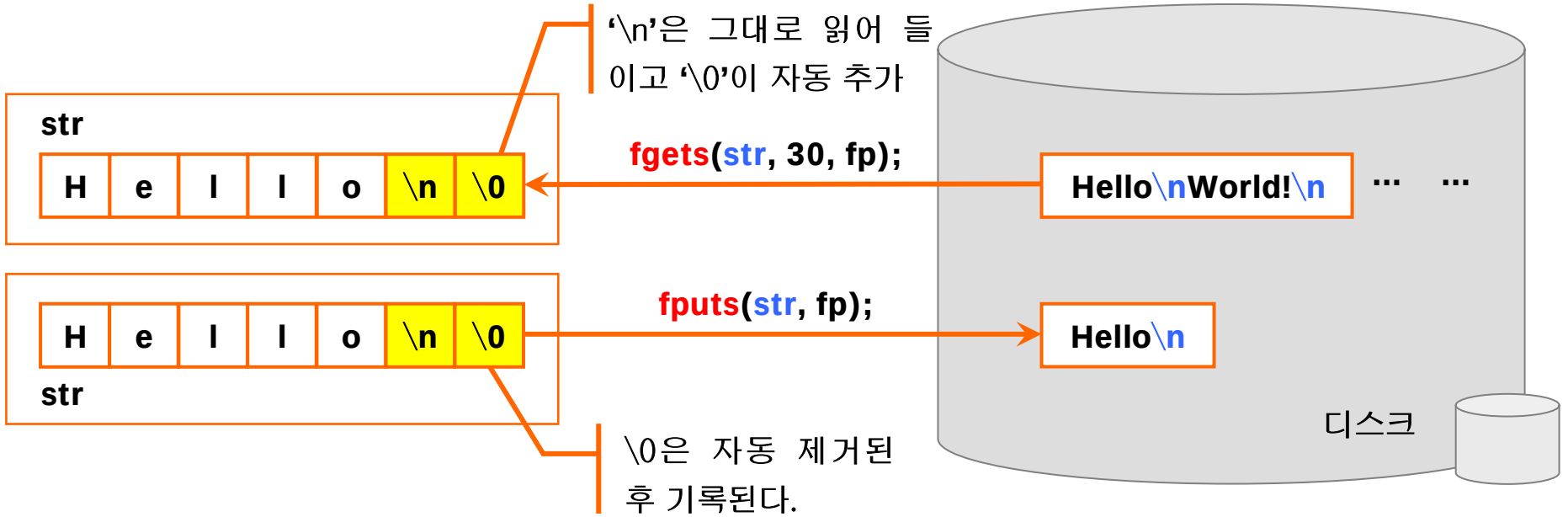
호출 성공 : 음수가 아닌 정수를 반환
호출 실패 : **EOF** 를 반환

  - str : 파일에 기록할 문자열
  - fp : 대상이 되는 파일 포인터

- str이 가리키는 문자열을 fp가 가리키는 대상 파일에 저장한다.

파일 입출력 함수 (cont'd)

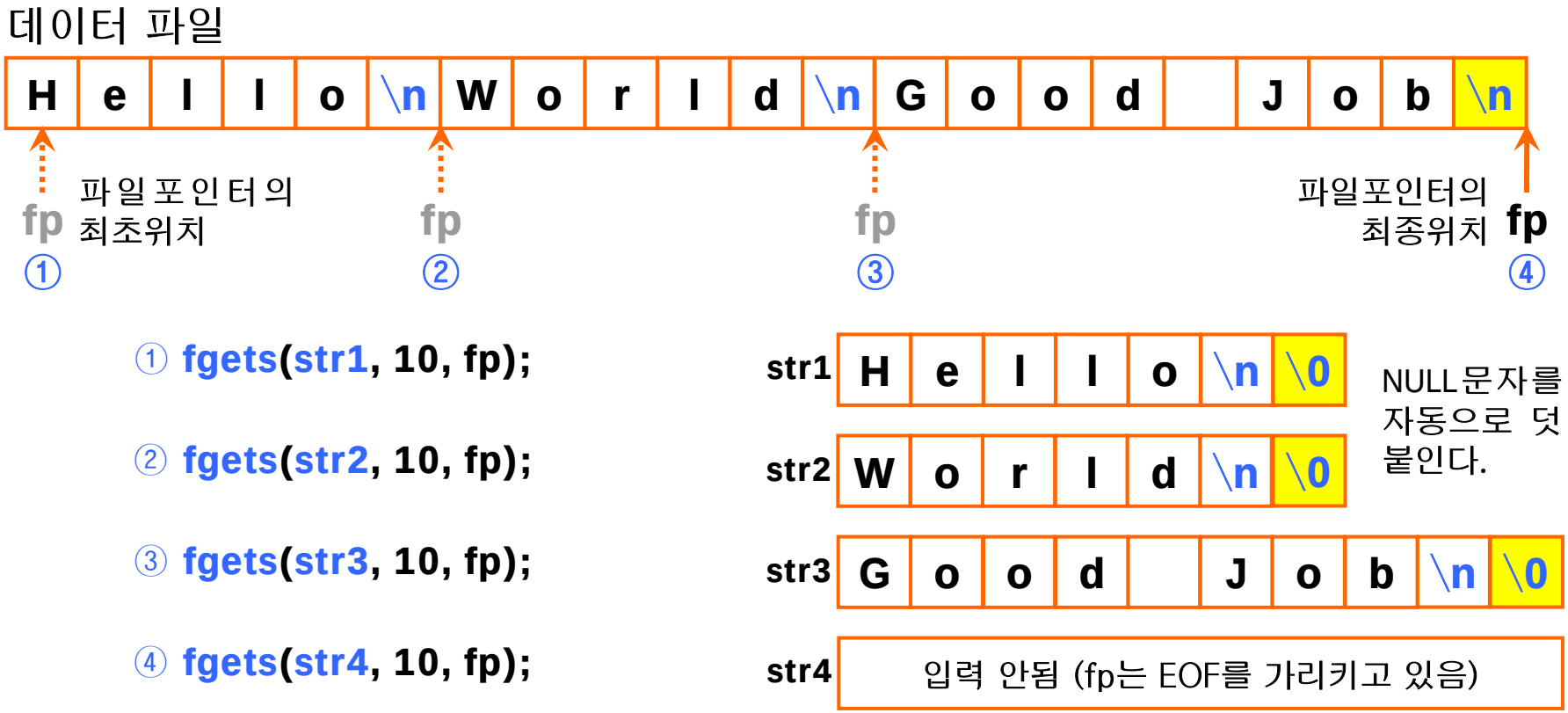
- **문자열 입출력 함수 (cont'd)**
  - fgets()와 fputs()의 기본 동작



파일 입출력 함수 (cont'd)

● 문자열 입출력 함수 (cont'd)

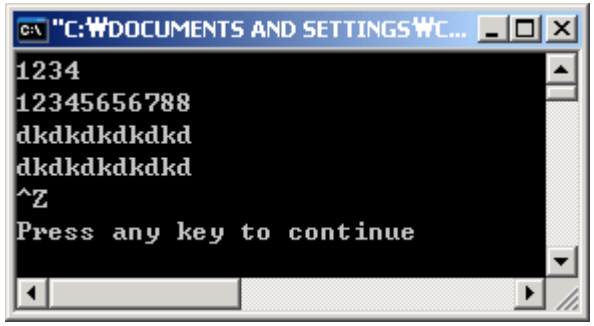
❍ fgets()의 동작

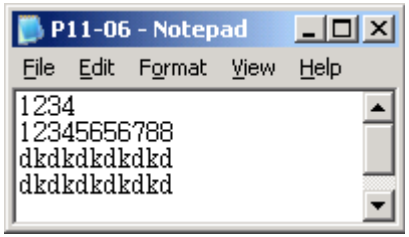


파일 입출력 함수 (cont'd)

프로그램 예제 : fgets () 와 fputs () 를 이용한 타자기 프로그램

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

int main (void)
{
        char    str[100];
        FILE    *fpOut;

        if (!(fpOut = fopen ("P11-06.TXT", "w")))  {
            printf("\aCould not open output file.\n");
            exit (100);
        }

        while ( fgets (str, sizeof (str), stdin) )
            fputs (str, fpOut);

        fclose (fpOut);

        return 0;
}
```





파일 입출력 함수 (cont'd)

● 서식화된 파일 입출력 함수 : fscanf()

❍ 파일을 대상으로 서식화된 입력 기능을 지원하는 함수

```
#include <stdio.h>

char  fscanf(FILE  *fp,  char  *format,  가변 길이 인수 리스트);
```

호출 성공 : 정수를 반환
호출 실패 : **EOF** 를 반환

- fp : 대상 파일 포인터
- format : 서식 문자열
- 가변 길이 인수 리스트 : 파일로부터 읽은 자료를 보관할 변수 목록

• "파일에서 읽어 들인 항목 수를 정수로 반환"

파일 입출력 함수 (cont'd)

● 서식화된 파일 입출력 함수 : fprintf()

❍ 파일을 대상으로 서식화된 출력 기능을 지원하는 함수

```
#include <stdio.h>

char  fprintf(FILE  *fp,  char  *format,  가변 길이 인수 리스트);
```

호출 성공 : 정수를 반환
호출 실패 : **EOF** 를 반환

- fp : 대상 파일 포인터
- format : 서식 문자열
- 가변 길이 인수 리스트 : 파일로 저장할 내용을 담고 있는 변수 목록

• "파일에 기록한 바이트 수를 정수로 반환"

파일 입출력 함수 (cont'd)

서식화된 파일 입출력 함수 (cont'd)

fscanf 사용 예

- 대쉬(-)로 구분된 데이터를 읽어서 각 부분을 3개의 정수 변수에 저장하는 fscanf() 문을 작성하라. 입력은 fpData로 열린 파일에서 읽는다.

```
5-10-1936
```

```
fscanf(fpData, "%d-%d-%d", &month, &day, &year);
```

- fpData로 열린 파일로부터 정수를 3개 읽고, 성공적으로 읽어서 변환된 변수의 수를 result라는 변수에 저장하는 fscanf() 문을 작성하라.

```
5       10      15
```

```
result = fscanf(fpData, "%d %d %d", &i, &j, &k);
```

파일 입출력 함수 (cont'd)

서식화된 파일 입출력 함수 (cont'd)

fscanf 사용 예 (cont'd)

- 한 라인에 정수가 네 개 있는 파일이 있을 때, 각 라인에서 첫 번째, 두 번째, 네 번째 정수만 읽는 fscanf() 문을 작성하라. 즉 세 번째 정수는 버린다.

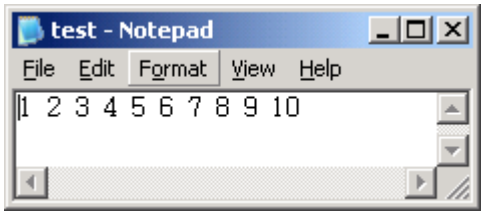
```
5       10      15      1
```

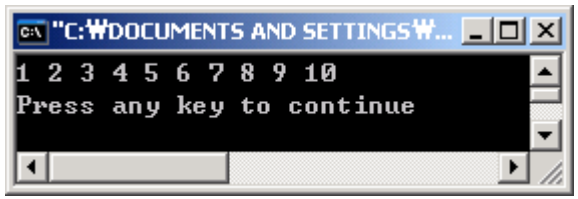
```
result = fscanf(fpData, "%d%d%*d%d", &a, &b, &d);
```

파일 입출력 함수 (cont'd)

프로그램 예제 : 정수가 저장된 텍스트 파일을 읽어서 출력하기

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

int main(void)
{
        FILE    *fpIn;
        int     numIn;

        fpIn = fopen("test.txt", "r");
        if(!fpIn)
        {
                printf("Could not open file \a\n");
                exit(101);
        }

        while ((fscanf(fpIn, "%d", &numIn)) == 1)
                printf("%d ", numIn);

        printf("\n");

        return 0;
}
```





파일 입출력 함수 (cont'd)

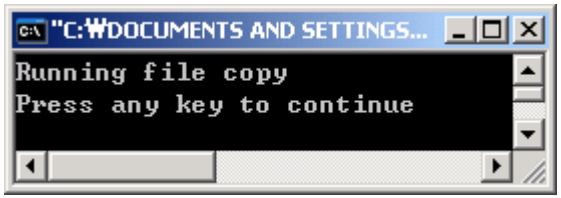
프로그램 예제 : 정수가 저장된 파일을 복사하기 (1/2)

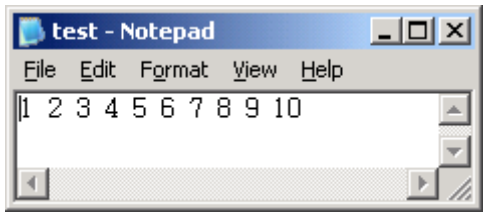
```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        FILE    *fpIn, *fpOut;
        int     numIn;
        int     closeResult;

        printf("Running file copy\n");

        fpIn = fopen("test.txt", "r");
        if(!fpIn)
        {
                printf("Could not open file \a\n");
                exit(101);
        }
```

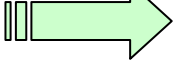
파일 입출력 함수 (cont'd)

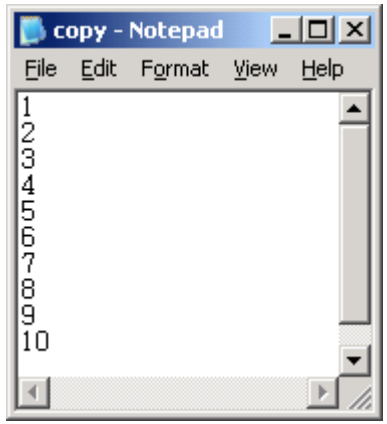
프로그램 예제 : 정수가 저장된 파일을 복사하기 (2/2)

```
        fpOut = fopen("copy.txt", "w");
        if(!fpOut)
        {
                printf("Could not open file \a\n");
                exit(102);
        }

        while ((fscanf(fpIn, "%d", &numIn)) == 1)
                fprintf(fpOut, "%d₩n", numIn);

        closeResult = fclose(fpOut);
        if(closeResult == EOF)
        {
                printf("Could not close output file \a\n"
                exit(201);
        }

        return 0;
}
```





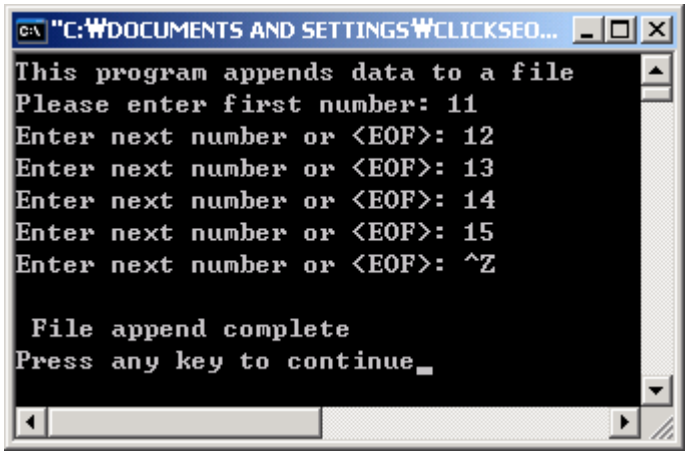


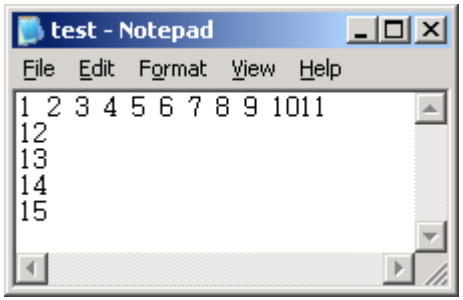


파일 입출력 함수 (cont'd)

프로그램 예제 : 파일에 추가하기 (1/2)

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

int main(void)
{
        FILE    *fpAppend;
        int     numIn;
        int     closeResult;

        printf("This program appends data to a file\n");

        fpAppend = fopen("test.txt", "a");
        if(!fpAppend)
        {
                printf("Could not open file \a\n");
                exit(101);
        }
```





파일 입출력 함수 (cont'd)

프로그램 예제 : 파일에 추가하기 (2/2)

```
        printf("Please enter first number: ");
        while((scanf("%d", &numIn)) != EOF)
        {
                fprintf(fpAppend, "%d₩n", numIn);
                printf("Enter next number or <EOF>: ");
        }

        closeResult = fclose(fpAppend);
        if(closeResult == EOF)
        {
                printf("Could not close output file \a\n");
                exit(201);
        }

        printf("\n File append complete\n");

        return 0;
}
```

파일 입출력 함수 (cont'd)

- **이진 파일 입출력 함수 : fread(), fwrite()**
  - 이진 모드는 입출력 작업 시 데이터 변환과정을 거치지 않는다.

  - fread() 함수
    - 이진 모드로 데이터를 읽는 함수

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

      - buffer : 파일로부터 읽어 들인 데이터를 기억시킬 버퍼를 가리키는 포인터
      - size : 한 번에 읽어 들일 수 있는 데이터의 바이트 수
      - n : size 만큼 읽어 들이기 위해 지정하는 반복 횟수
      - fp : 대상이 되는 파일 포인터

      - "읽어 들인 블록(size의 크기를 하나의 데이터 블록)의 개수를 반환"
      - 반환 값 : 0
        - » 파일의 끝을 만나 더 이상 읽어 들일 자료가 없거나 오류가 발생한 경우이다.

파일 입출력 함수 (cont'd)

이진 파일 입출력 함수 (cont'd)

fwrite() 함수

- 이진 모드로 데이터를 쓰는 함수

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

- buffer : 파일에 기록하고자 하는 데이터가 들어 있는 버퍼로의 포인터
- size : 한 번에 기록할 데이터의 바이트 수
- n : size 만큼 쓰기 위해 지정하는 반복 횟수
- fp : 대상이 되는 파일 포인터

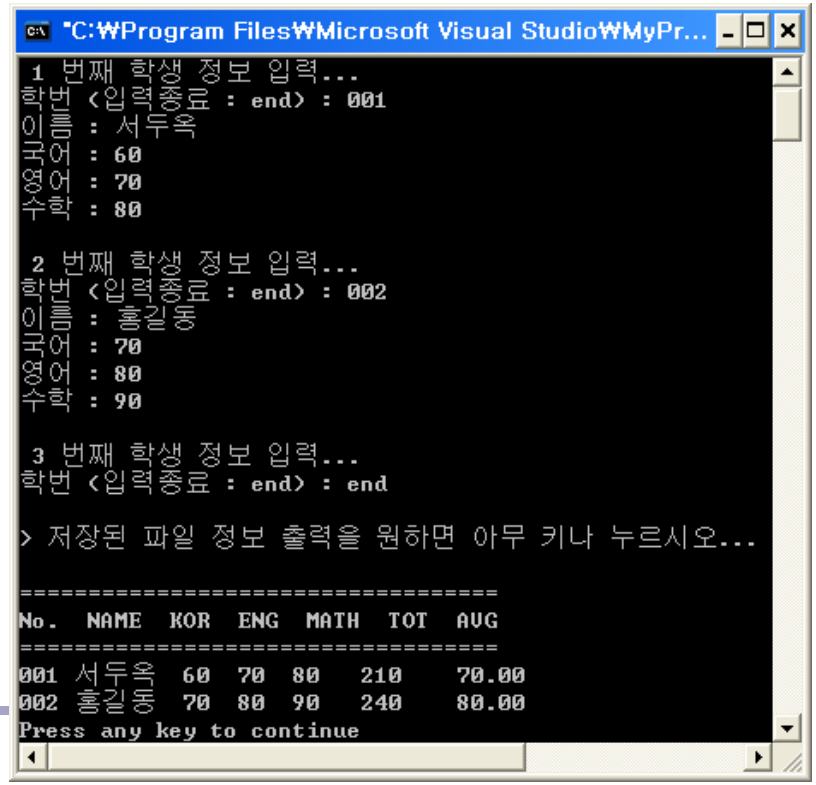
- "자신이 기록한 블록(size의 크기를 하나의 데이터 블록)의 개수를 반환"

- "반환 값이 n이 아니라면 기록 도중 오류가 발생한 경우"

파일 입출력 함수 (cont'd)

프로그램 예제 : fread()와 fwrite()를 이용한 블록 입출력 예제 (1/3)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

typedef struct student_score // 블록 단위 입출력을 위한 성적 데이터
{
        char    no[10];
        char    name[20];
        int     kor, eng, math, tot;
        double  avg;
} INFO;

char *fname = "result.txt";

int main()
{
        int     cnt = 0;
        FILE    *fp;
        INFO    temp;
```



파일 입출력 함수 (cont'd)

프로그램 예제 : fread()와 fwrite()를 이용한 블록 입출력 예제 (2/3)

```
// 이진 모드로 파일 개방
if ((fp = fopen(fname,"wb") ) == NULL )    {
        printf("File Open Error!\n");
        exit(1);
}

while (1)  {
        printf("\n %d 번째 학생 정보 입력...\n", ++cnt);
        printf("학번 (입력종료 : end) : ");
        gets(temp.no);
        if(!strcmp(temp.no, "end"))
                break;
        printf("이름 : ");      gets(temp.name);
        printf("국어 : ");      scanf("%d", &temp.kor);
        printf("영어 : ");      scanf("%d", &temp.eng);
        printf("수학 : ");      scanf("%d%*c", &temp.math);

        temp.tot = temp.kor + temp.eng + temp.math;
        temp.avg = temp.tot / 3.0;

        // 기록할 때는 구조체 크기만큼 한꺼번에 기록한다.
        fwrite(&temp, sizeof(INFO), 1, fp);
}
fclose(fp);
```

파일 입출력 함수 (cont'd)

프로그램 예제 : fread()와 fwrite()를 이용한 블록 입출력 예제 (3/3)

```
        printf("\n> 저장된 파일 정보 출력을 원하면 아무 키나 누르시오... ");
        getch();
        // 읽어 들일 때도 이진 모드로 파일 개방
        if ((fp = fopen(fname,"rb") ) == NULL) {
                printf("File Open Error!\n");
                exit(1);
        }

        printf("\n\n=====================================\n");
        printf("No.  NAME  KOR  ENG  MATH  TOT  AVG\n");
        printf("=====================================\n");
        while (1)  {   // 구조체 temp 크기만큼 읽어 온다.
                cnt = fread(&temp, sizeof(INFO), 1, fp);
                if (cnt == 0 || cnt == EOF)  break;

                printf("%s %s %3d %3d %3d %5d %8.2f\n",
                        temp.no, temp.name, temp.kor, temp.eng, temp.math,
                        temp.tot, temp.avg);
        }

        fclose(fp);
        return 0;
}
```

파일 입출력 함수 (cont'd)

랜덤 액세스 함수 : fseek(), ftell()

fseek() 함수

- 파일 포인터 fp를 임의의 위치로 이동하는 함수
- 원하는 자료에 대한 직접 접근(direct access)이 가능

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

- fp : 대상이 되는 파일 포인터
- offset : whence 위치부터 새로운 위치까지 상대적으로 떨어진 거리 (바이트 단위)
- whence : 파일 포인터 이동을 위한 기준점

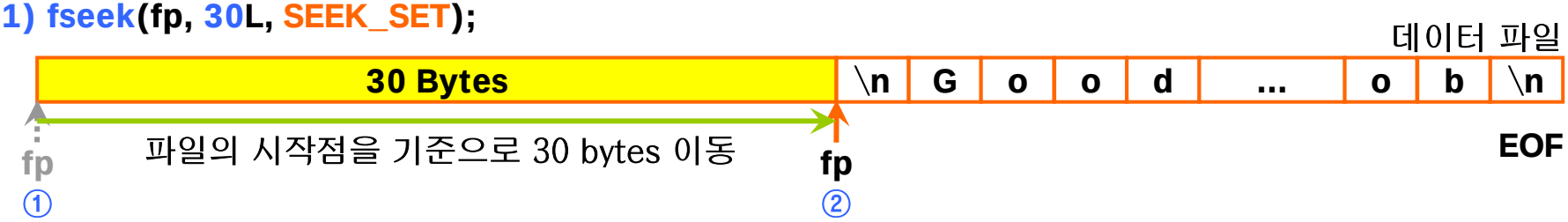
whence	값	의미
SEEK_SET	0	파일의 시작위치를 기준으로 파일포인터를 이동하겠다는 뜻
SEEK_CUR	1	현재의 파일포인터 위치를 기준으로 다음 위치로 이동시키겠다는 뜻
SEEK_END	2	파일의 마지막 위치를 기준으로 파일포인터를 이동하겠다는 뜻

파일 입출력 함수 (cont'd)

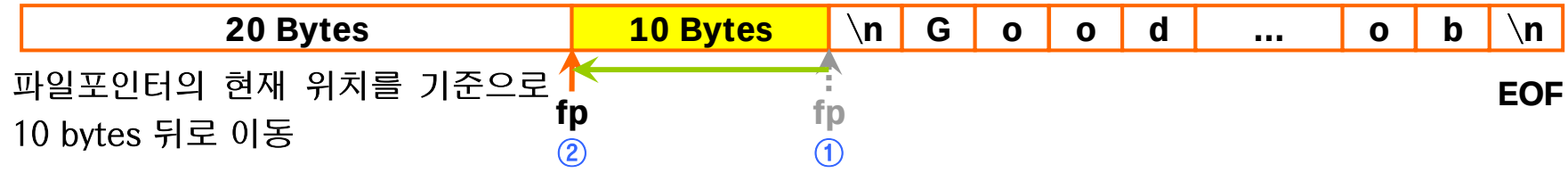
● 랜덤 액세스 함수 (cont'd)

❍ fseek() 함수 동작 원리

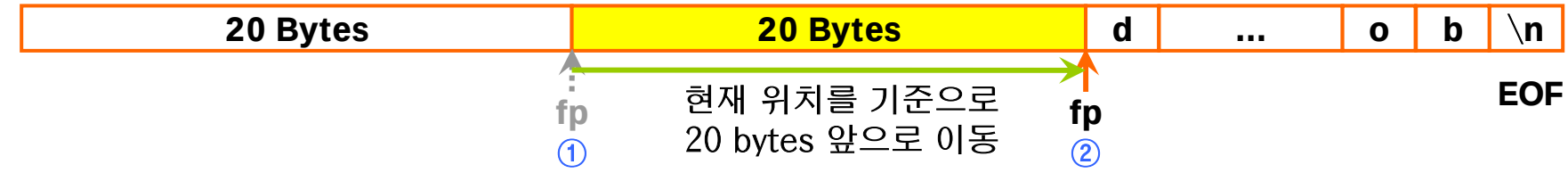
1) fseek(fp, 30L, SEEK_SET);



2) fseek(fp, -10L, SEEK_CUR);



3) fseek(fp, 20L, SEEK_CUR);

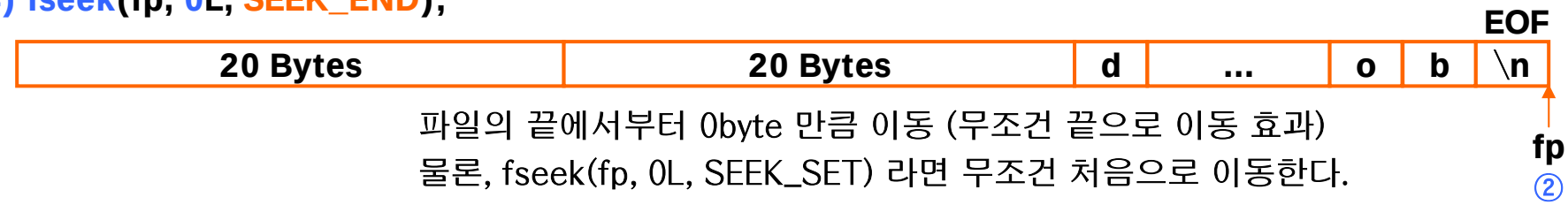


파일 입출력 함수 (cont'd)

● 랜덤 액세스 함수 (cont'd)

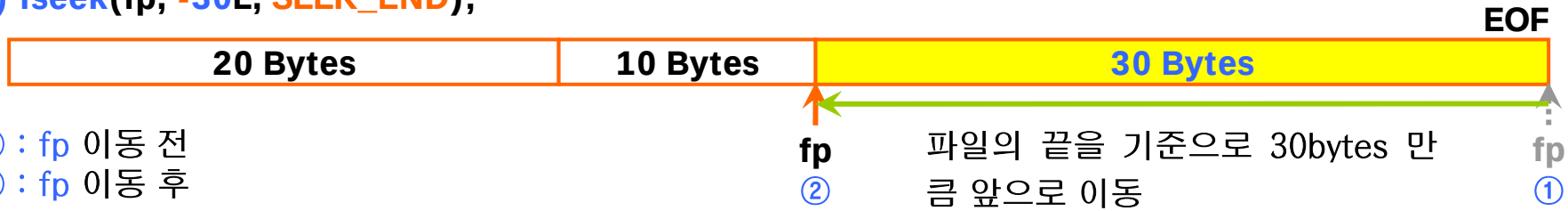
❍ fseek() 함수 동작 원리 (cont'd)

4) fseek(fp, 0L, SEEK_END);



파일의 끝에서부터 0byte 만큼 이동 (무조건 끝으로 이동 효과)
물론, fseek(fp, 0L, SEEK_SET) 라면 무조건 처음으로 이동한다.

5) fseek(fp, -30L, SEEK_END);



① : fp 이동 전
② : fp 이동 후

파일의 끝을 기준으로 30bytes 만큼 앞으로 이동

파일 입출력 함수 (cont'd)

랜덤 액세스 함수 (cont'd)

ftell() 함수

- 파일 포인터의 위치를 얻어오는 함수
- 첫 시작 위치부터 현재 fp 위치까지의 거리를 long 형으로 알려준다.
  - "오류가 발생하면 -1을 반환"

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

  - fp : 대상이 되는 파일 포인터

기타 파일 처리 함수

- 파일 입출력 개념
- 파일 입출력 함수
- **기타 파일 처리 함수**
  - **오류 처리 관련 함수**
  - **기타 함수**

기타 파일 처리 함수

● 오류 처리 관련 함수 : feof()

❍ feof() 함수

- 파일의 끝에 도달했는지 여부를 알려주는 함수
  - 파일의 끝을 읽으면 0이 아닌 수를 반환하고, 그렇지 않은 경우는 0을 반환한다.

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

  - fp : 대상이 되는 파일 포인터

- feof()함수의 일반적인 사용 형식

```
while (!feof(fp))                  // 파일의 끝이 아니라면
        putchar(fgetc(fp));  // 한 문자씩 읽어서 화면에 바로 출력
```

기타 파일 처리 함수 (cont'd)

● 오류 처리 관련 함수 : ferror()

❍ ferror() 함수

- 파일 처리 도중 입출력 오류가 발생할 때 오류를 알려주는 함수
  - 파일 포인터 fp와 관련된 처리에 입출력 오류가 없으면 0을 반환하고, 그렇지 않으면 0이 아닌 값을 반환하여 오류를 알린다.
  - 파일 처리 도중에 발생할 수 있는 상태
    - » 읽기 오류(read error)
    - » 쓰기 오류(write error)
    - » 파일 끝(end of file)

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

  - fp : 대상이 되는 파일 포인터

기타 파일 처리 함수 (cont'd)

● 오류 처리 관련 함수 : clearerr()

❍ clearerr() 함수

- error 상태를 clear하는 함수
  - FILE 구조체의 멤버 중 flag의 EOF Indicator 비트를 0으로 설정하고, ferror()가 설정해 놓은 Error Indicator 비트도 0으로 만들어 오류 상태를 되돌려 놓는 역할을 한다.

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

  - fp : 대상이 되는 파일 포인터

기타 파일 처리 함수 (cont'd)

오류 처리 관련 함수 : perror()

perror() 함수

- 입출력 오류가 발생할 경우에 오류 메시지를 콘솔로 출력하는 함수
  - 파일이 없거나 이미 존재하는 경우 등 다수의 상황에 따른 오류 코드를 **error.h**에 매크로 상수로 정의해 두는데, 오류가 발생할 경우 **errno**에 자동으로 저장된다.

```
void      perror(const      char *errmsg);
```

  - errmsg : 출력하려는 오류 메시지

```
#include <errno.h>
...
fp = fopen("perror.txt", "r");
if (!fp) {
        perror("Unable to open file for reading");
        printf("errno = %d\n", errno);
}
```

기타 파일 처리 함수 (cont'd)

- **기타 함수 : fflush()**
  - fflush() 함수
    - 대상이 되는 파일 버퍼를 비우는 함수

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

  - fp : 대상이 되는 파일 포인터
  - fp가 stdout인 경우는 입출력 버퍼를 비우고, fp가 디스크 파일 스트림인 경우는 파일 버퍼를 비움으로써 버퍼의 내용을 최종적으로 디스크에 기록한다.
  - "작업이 성공적일 때는 0을 반환하고, 실패한다면 EOF를 반환"
  - 만약 fp가 NULL인 경우라면 모든 디스크 버퍼를 비운다.

응용 실습 1

- 입력 파일 내용을 읽어서 입력 파일과 다른 파일에 출력하는 파일 copy 프로그램을 작성하시오.

- 2개의 파일을 1개의 파일로 병합(Merge)하는 프로그램을 작성하시오.

- 확장자가 .c인 소스 파일을 open하여 그 파일 내의 중괄호가 있는 행만 행 번호를 추가 한 후 화면에 표시하는 프로그램을 작성하시오.

응용 실습 2

- **사용자로부터 파일 명을 입력 받아 line번호가 부여된 상태로 아래 출력 형식과 같이 표준출력 시키는 프로그램을 작성하시오.**
  - 단, line번호는 3자리로 하고 빈 line에 대해서는 line번호를 붙이지 않는다.

출력결과

```
001 #include <stdio.h>
002 #include <stdlib.h>

003 void main()
004 {
005         int I, j;
006         FILE *fp;
 ...         ......
```

응용 실습 3

- **입력 파일과 출력 결과를 참조해서 파일을 읽어 들여 알파벳의 빈도수(대/소문자 구별 없이)와 각 단어의 빈도수(정렬된 상태로)를 나타내는 프로그램을 작성하시오.**

입력파일

```
The detailed code example will get you
up-and-running with this newest technology for
P2P software development this code is
```

출력결과

알파벳	빈도수
A	4
B	0
C	3
…	…
Z	0

단어	빈도수
code	2
detailed	1
for	1
…	…
the	1

참고문헌

[1] 김일광 저, “C프로그래밍 입문”, 한빛미디어, pp. 356 – 423.

[2] 윤성우, “열혈강의 C 프로그래밍”, 프리렉, pp. 537 – 568.

[3] 김진 외 7인 공역, “구조적 프로그래밍 기법을 위한 C”, 인터비젼, pp. 358 – 416, pp.772 – 832.

[4] Brian W. Kernighan, Dennis M. Ritchie, “The C Programming Language” 2/e, 대영사, 206 – 229.