



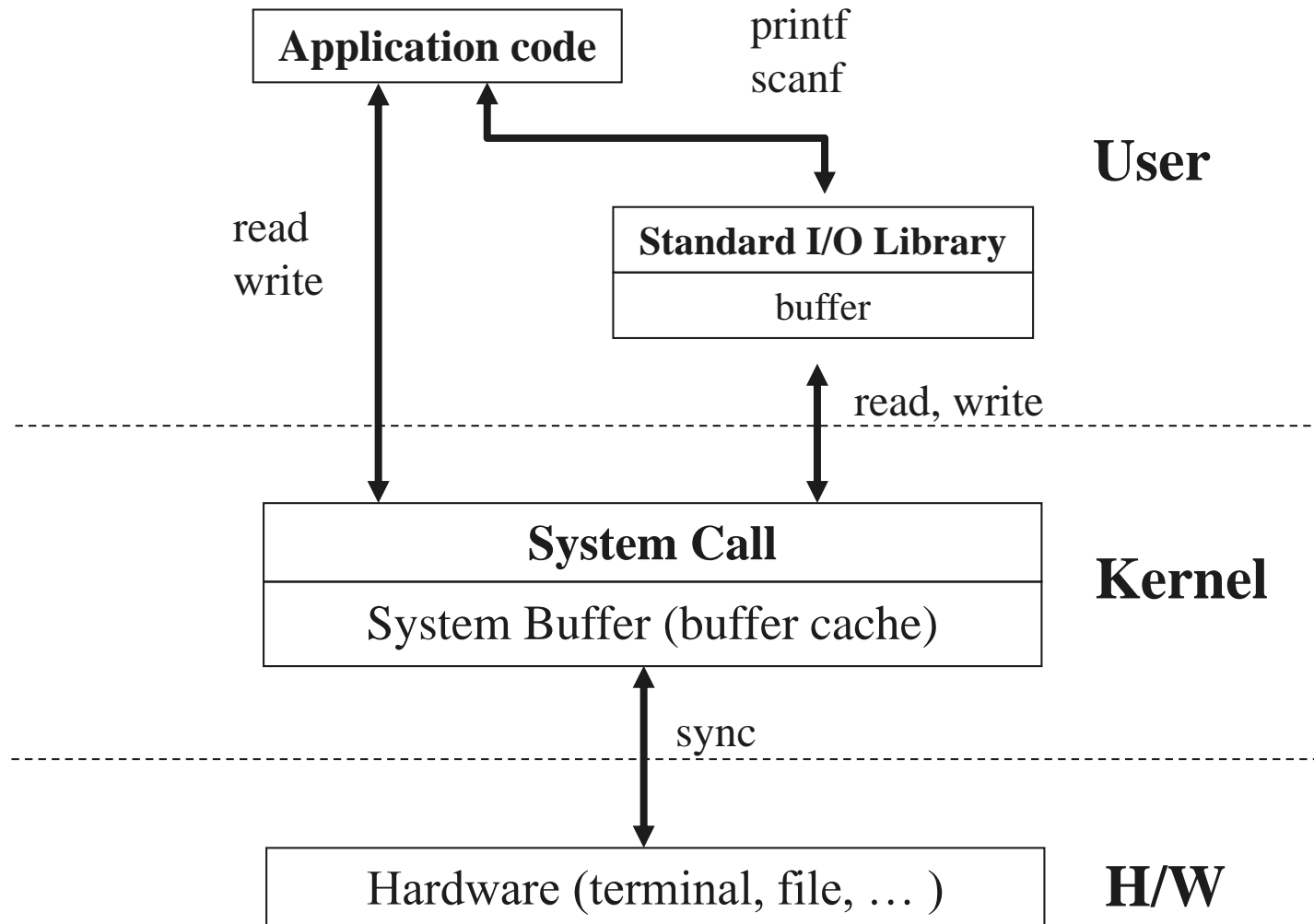
표준 입출력 라이브러리

로봇SW 교육원

최상훈(shchoi82@gmail.com)

시스템 호출 vs 표준 I/O 라이브러리 함수

2



스트림과 FILE 구조체

3

- 스트림 (stream)
 - 표준 I/O 라이브러리는 스트림을 사용하여 파일 입출력을 처리함
 - 파일을 생성하거나 열면 파일 스트림을 얻게 됨
 - 이것을 “스트림을 파일과 연관 시켰다” 라고 말함
- FILE 구조체
 - 표준 I/O 라이브러리의 스트림을 표현하는 구조체
 - 특정 파일과 연결
 - 구조체 내부에 파일 디스크립터를 포함
 - 모든 프로세스는 세가지 기본 표준 스트림 제공
 - 표준입력, 표준출력, 표준에러에 대한 스트림
 - stdin, stdout, stderr

스트림과 FILE 구조체

4

```
/usr/include/stdio.h
```

```
49 typedef struct _IO_FILE FILE;
```

※ <libio.h> 에 _IO_FILE이 정의되어 있음

```
/usr/include/stdio.h
```

```
164 /* Standard streams. */
```

```
165 extern struct _IO_FILE *stdin; /* Standard input stream. */
```

```
166 extern struct _IO_FILE *stdout; /* Standard output stream. */
```

```
167 extern struct _IO_FILE *stderr; /* Standard error output stream. */
```

```
/usr/include/unistd.h
```

```
210 /* Standard file descriptors. */
```

```
211 #define STDIN_FILENO 0 /* Standard input. */
```

```
212 #define STDOUT_FILENO 1 /* Standard output. */
```

```
213 #define STDERR_FILENO 2 /* Standard error output. */
```

실습1:스트림

5

- stdin, stdout, stderr 스트림의 파일 디스크립터 확인하기
- 파일명 : stdStream.c

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

int main(void)
{
    printf("stdin._fileno:%d\n", stdin->_fileno);
    printf("stdout._fileno:%d\n", stdout->_fileno);
    printf("stderr._fileno:%d\n", stderr->_fileno);
    return 0;
}
```

```
$ ./stdStream
stdin._fileno:0
stdout._fileno:1
stderr._fileno:2
$
```

버퍼링

6

- 프로그래머에게 효율적이고 편리한 프로그밍 환경 제공
 - 최적의 버퍼크기를 라이브러리가 자동으로 할당
 - read(), write() 를 최소화함으로써 성능 향상
- 버퍼링 방식
 - 전체버퍼링
 - 줄 단위 버퍼링
 - 버퍼링 없음
- 표준 스트림의 버퍼링 방식
 - 표준 오류는 항상 버퍼링 되지 않음
 - 표준 입력과 표준출력은 일반적으로 터미널 장치를 가리키는 경우에만 줄단위 버퍼링이 적용되고 그외에는 전체버퍼링됨

실습2:버퍼링 예제(1/2)

7

- 스트림의 버퍼링 방식 확인하기
- 파일명 : buffering.c

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

void pr_stdio(const char *, FILE *);

int main(void)
{
    FILE *fp;
    fputs("enter any character\n", stdout);
    if(getchar() == EOF){
        fprintf(stderr, "getchar error=\n");
        exit(1);
    }
    fputs("one line to standard erro\n", stderr);
    pr_stdio("stdin", stdin);
    pr_stdio("stdout", stdout);
    pr_stdio("stderr", stderr);

    if((fp = fopen("/etc/motd", "r")) == NULL){
        fprintf(stderr, "fopen error\n");
        exit(1);
    }
}
```

실습2:버퍼링 예제(2/2)

8

```

    if(getc(fp) == EOF){
        fprintf(stderr, "getc error\n");
        exit(1);
    }
    pr_stdio("/etc/motd", fp);
    exit(0);
}

void pr_stdio(const char *name, FILE *fp)
{
    printf("stream = %s, ", name);
    if(fp->_IO_file_flags & _IO_UNBUFFERED)
        printf("unbuffered");
    else if(fp->_IO_file_flags & _IO_LINE_BUF)
        printf("line buffered");
    else printf("fully buffered");
    printf(", buffered size = %d\n", fp->_IO_buf_end - fp->_IO_buf_base);
}

```

```

$ ./buffering
enter any character
[enter]
one line to standard erro
stream = stdin, line buffered, buffered size = 1024
stream = stdout, line buffered, buffered size = 1024
stream = stderr, unbuffered, buffered size = 1
stream = /etc/motd, fully buffered, buffered size = 4096
$

```


setbuf, setvbuf

9

```
#include <stdio.h>

void setbuf (FILE *fp, char *buf );

int setvbuf (FILE *fp, char *buf, int mode, size_t size );
```

- 버퍼의 관리 기법을 변경
- 스트림에 대해 다른 입출력 연산 수행되기 전에 호출
- setbuf ()
 - 버퍼 사용을 켜거나 끌 수 있음
 - *buf* 를 NULL로 설정하면 버퍼를 사용하지 않겠다는 의미
 - 버퍼를 사용하기 위해서는, BUFSIZ크기의 버퍼사용
 - setvbuf(fp, buf, _IOFBF, BUFSIZ) 와 동일

실습3:setbuf 예제

10

- 줄 단위 버퍼링
- 파일명 : setbuf.c

```
#include<stdio.h>
#include<unistd.h>
int main(void)
{
    char buf[BUFSIZ];
    setbuf(stdout,buf);
    printf("Hello, ");           sleep(1);
    printf("UNIX!");           sleep(1);
    printf("\n");               sleep(1);
    setbuf(stdout,NULL);
    sleep(1);
    printf("How ");             sleep(1);
    printf("are ");             sleep(1);
    printf("you?");             sleep(1);
    printf("\n");
    return 0;
}
```

```
$ ./setbuf
Hello, UNIX!
How are you?
$
```

setbuf, setvbuf

11

- setvbuf ()
 - 버퍼 사용 방법을 변경
 - *mode*
 - _IOFBF : 전체버퍼링
 - _IOLBF : 라인버퍼링
 - _IONBF : 버퍼링없음
 - *buf*
 - mode 가 _IONBF이면 무시됨
 - 버퍼의 주소
 - NULL이면 표준I/O라이브러리가 적절한 크기(BUFSIZ)로 직접할당
 - *size*
 - mode 가 _IONBF이면 무시됨
 - 버퍼의 크기

실습4:버퍼크기

12

- BUFSIZ와 st_blksize
- 파일명 : bufsiz.c

```
#include<stdio.h>
#include<sys/stat.h>

int
main(void)
{
    struct stat statbuf;
    printf("BUFSIZ:%d\n", BUFSIZ);
    if(stat(".", &statbuf) == -1){
        fprintf(stderr, "stat error\n");
        return 1;
    }
    printf("st_blksize:%ld\n", statbuf.st_blksize);
    return 0;
}
```

```
$ ./bufsiz
BUFSIZ:8192
st_blksize:4096
$
```

fflush()

13

```
#include <stdio.h>

int fflush (FILE *fp);
```

- 기능: 명시적으로 버퍼 방출
- 리턴 값 : 성공하면 0, 실패하면 EOF (-1)
- *fp* 에 NULL을 설정하면, 스트림들의 출력버퍼가 전부 방출됨

fopen

14

```
#include <stdio.h>

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

- **기능: 해당 파일의 스트림을 연다**
- **리턴 값 : 성공하면 FILE 포인터, 실패하면 NULL**
- *type*
 - r , rb : O_RDONLY
 - w , wb : O_WRONLY | O_CREAT | O_TRUNC
 - a , ab : O_WRONLY | O_CREAT | O_APPEND
 - r+ , r+b , rb+ : O_RDWR
 - w+ , w+b , wb+ : O_RDWR | O_CREAT | O_TRUNC
 - a+ , a+b , ab+ : O_RDWR | O_CREAT | O_APPEND
- **기본적으로 전체 버퍼링 방식이 적용됨**
 - 단, 터미널에 대한 스트림일경우 줄단위 버퍼링 적용

fopen

15

제약	r	w	a	r+	w+	a+
파일이 반드시 존재해야 함 파일의 이전 내용이 폐기됨	○	○		○	○	
스트림을 읽을 수 있음 스트림을 쓸 수 있음 스트림의 끝에서만 쓸 수 있음	○	○	○ ○	○ ○	○ ○	○ ○ ○

실습5:fopen 예제

16

- "a+" 파일스트림 열기
- 파일명 : fopen.c

```
#include <stdio.h>

int main(void)
{
    FILE    *fp;

    if((fp = fopen("./test", "a+")) == NULL) {
        fprintf(stderr, "Error\n");
        return 0;
    }
    printf("Success !\n");
    printf("fd:%d\n", fp->_fileno);
    fclose(fp);

    return 0;
}
```

```
$ ./fopen
Success !
fd:3
$
```


freopen, fdopen

17

```
#include <stdio.h>

FILE *freopen (const char *pathname, const char *type, FILE *fp );

FILE *fdopen (int fildes, const char *type);
```

- **freopen ()**
 - **기능: 지정된 파일을 지정된 스트림으로 연다**
 - **리턴 값 : 성공하면 FILE 포인터, 실패하면 NULL**
 - **스트림이 이미 열려있으면 닫고 지정된파일로 스트림을 다시 연다**
- **fdopen ()**
 - **기능 : 이미 열려진 파일 디스크립터 (fildes)에 대해 스트림과 연관시킴**
 - **리턴 값 : 성공하면 FILE 포인터, 실패하면 NULL**
 - **open, dup, dup2, fcntl 등의 함수로 얻은 파일 디스크립터를 사용**

실습6:freopen 예제

18

- freopen 파일스트림 열기
- 파일명 : freopen.c

```
#include <stdio.h>

int main(void)
{
    char *fname = "test";
    FILE *fp;
    printf("First printf is on the screen.\n");
    if((fp = freopen(fname, "w", stdout)) == NULL){
        fprintf(stderr, "freopen\n");
        return 1;
    }
    printf("Second printf is in this file.\n");
    return 0;
}
```

```
$ ./freopen
First printf is on the screen.
$ cat test
Second printf is in this file.
$
```

fclose

19

```
#include <stdio.h>

int fclose ( FILE *fp );
```

- **기능: 스트림을 닫음**
- **리턴 값: 성공하면 0, 실패하면 EOF**
- **버퍼에 있는 출력 자료가 방출되고 버퍼에 있는 입력자료는 폐기됨**
- **버퍼를 스스로 할당했었다면 버퍼가 해제됨**
- **프로세스가 정상적으로 종료되면 모든 I/O 스트림 버퍼에 있는 자료가 방출되고 스트림들이 모두 닫힘**

입출력 함수의 종류

20

- **문자 단위 I/O**
 - getc, fgetc, getchar, ungetc
 - putc, fputc, putchar
- **줄 단위 I/O**
 - gets, fgets
 - puts, fputs
- **이진(binary) I/O**
 - fread, fwrite
- **서식화된 (formatted) I/O**
 - scanf, fscanf, sscanf
 - printf, fprintf, sprintf

getc, fgetc, getchar

21

```
#include <stdio.h>

int getc (FILE *fp );
int fgetc (FILE *fp );
int getchar (void );
```

- **기능:** 스트림에서 한 문자를 읽어 오는 함수
- **리턴 값:** 성공하면 읽은 문자, 실패하거나 파일의 끝이면 EOF
- **getchar**은 **표준입력(stdin)**스트림으로부터 문자 하나 입력 받음
- **fgetc(stdin),getc(stdin)** 와 **getchar()** 동일함

ungetc

22

```
#include <stdio.h>

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

- 기능: 읽은 문자를 다시 스트림에게 되돌려 놓음
- 리턴 값: 성공하면 *c*, 실패하면 EOF

실습7:fgetc, ungetc 예제

23

- ungetc 함수 사용하기
- 파일명 : ungetc.c

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

int main(void)
{
    FILE *fp;
    int c;
    if((fp = fopen("test", "r")) == NULL) {
        fprintf(stderr, "fopen error\n");
        return 1;
    }
    c = fgetc(fp);
    printf("%c", c);
    c = fgetc(fp);
    printf("%c", c);
    c = fgetc(fp);
    printf("%c", c);

    ungetc(c, fp);

    c = fgetc(fp);
    printf("%c", c);

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

```
$ cat > test
abcdefg
[Ctrl+d]
$ ./ungetc
abcc$
```

error, feof, clearerr

24

```
#include <stdio.h>

int ferror (FILE *fp);

int feof (FILE *fp);

void clearerr (FILE *fp);
```

- ferror
 - 오류가 발생했는지 확인
- feof
 - 파일 끝에 도달했는지 확인
- FILE 구조체안의 플래그
 - 오류 플래그
 - 파일 끝 플래그
- clearerr는 오류 플래그 와 파일 끝 플래그를 모두 해제함

실습8:feof, ferror 예제

25

- feof와 ferror
- 파일명 : feof.c

```
#include<stdio.h>

int main(void)
{
    int c;
    while((c = getc(stdin)) != EOF)
        if(putc(c, stdout) == EOF){
            printf("output error\n");
            return 1;
        }
    if(ferror(stdin)){
        printf("input error\n");
        return 1;
    }
    if(feof(stdin)){
        printf("input eof\n");
        return 1;
    }
    return 0;
}
```

```
$ ./feof
hello advc
hello advc
[Ctrl+d]input eof
$ cat > file1
hello advc
[Ctrl+d]
$ ./ feof > file2 < file1
$ cat file2
hello advc
input eof
$
```

putc, fputc, putchar

26

```
#include <stdio.h>

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

- **기능 : 스트림에 한 문자를 출력함**
- **리턴 값 : 성공하면 c, 실패하면 EOF**
- **putchar은 표준출력(stdout)으로 출력함**
- **putchar(c) 와 fputc(c, stdout) 동일함**

fgets, gets

27

```
#include <stdio.h>

char *fgets (char *buf, int n, FILE *fp );

char *gets (char *buf );
```

- fgets
 - 스트림에서 n 만큼 한줄을 읽음
 - 새줄 문자까지 읽거나, n-1 개 읽어 buf에 저장함
 - buf 널처리됨
- gets
 - 표준입력에서 한줄을 읽음

fputs, puts

28

```
#include <stdio.h>

int fputs (const char *str, FILE *fp );

int puts (const char *str);
```

- fputs
 - 스트림에 str(문자열)을 출력함
 - 마지막 문자가 반드시 새줄 문자 아니어도 됨
- puts
 - 표준출력으로 str(문자열)을 출력함

fread, fwrite

29

```
#include <stdio.h>

size_t fread (void *ptr, size_t size, size_t nobj, FILE *fp );
size_t fwrite (const void *ptr, size_t size, size_t nobj, FILE *fp );
```

- 기능: 스트림에(서) 이진 자료를 쓴다(읽는다)
- 리턴 값: 입출력 객체의 개수
- *ptr*: 이진자료의 주소
- *size*: 원소의 크기
- *nobj*: 원소의 개수
- fread함수는 오류가 발생했거나 파일끝에 도달했다면 *nobj*보다 작은 값을 리턴, *ferror*또는 *feof*로 알아내야 함

실습9:fread, fwrite 예제1

30

- fwrite를 이용한 파일 쓰기
- 파일명 : fwrite.c

```
#include<stdio.h>

int main(void)
{
    char data[10] = {'a','b','c','d','e','h'};
    FILE *fp;

    if((fp = fopen("./test3", "w")) == NULL){
        fprintf(stderr, "fopen error\n");
        return 1;
    }
    if(fwrite(data, sizeof(char), 4, fp) != 4){
        fprintf(stderr, "fwrite error\n");
        return 1;
    }

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

```
$ ./fwrite
$ cat test3
abcd$
```

실습10:fread, fwrite 예제2

31

- 구조체정보 파일로 쓰기
- 파일명 : fwrite2.c

```
#include<stdio.h>

typedef struct {
    char name[10];
    long total;
}ITEM;

int main(void)
{
    ITEM items[2] = {"shchoi82",20},{"adv",30};
    ITEM items2[2];
    FILE *fp,*fp2;
    int nread;
    int i;
    if((fp = fopen("./test4", "w")) == NULL){
        fprintf(stderr, "fopen error\n");
        return 1;
    }
    if(fwrite(items, sizeof(ITEM), 2, fp) != 2){
        fprintf(stderr, "fwrite error\n");
        return 1;
    }
    fclose(fp);
```

실습10:fread, fwrite 예제2

32

```
if((fp2 = fopen("./test4", "r")) == NULL){
    fprintf(stderr, "fopen error\n");
    return 1;
}
nread = fread(items2, sizeof(ITEM), 2, fp2);
printf("nread:%d\n", nread);
for(i = 0 ; i < nread ; i++){
    printf("name : %s\n", items2[i].name);
    printf("total : %ld\n", items2[i].total);
}
fclose(fp2);

return 0;
}
```

```
$ ./fwrite2
nread:2
name : shchoi82
total : 20
name : advc
total : 30
$
```


Binary I/O

33

- Binary I/O의 문제점
 - 데이터를 쓴 시스템과 읽는 시스템이 다를 경우 문제가 발생할 수 있음
 - 구조체 내의 각 필드의 오프셋은 컴파일러와 시스템 환경에 따라 다를 수 있음
 - 동일 시스템 환경에서도 컴파일 옵션에 따라서 오프셋이 다를 수 있음
 - 데이터 형에 대해서 각 바이트의 순서가 시스템마다 다를 수 있음
Little Endian, Big Endian
 - 해결 방법
 - 고수준의 프로토콜에 의해서 데이터의 변환을 수행한다.

ftell, fseek, rewind

34

```
#include <stdio.h>

long  ftell (FILE *fp );

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

void  rewind (FILE *fp);
```

- ftell()
 - 파일의 현재 오프셋을 돌려 줌
- fseek()
 - 기능: 파일의 현재 파일 오프셋을 변경
 - 리턴 값: 성공하면 0, 실패하면 -1
 - whence : lseek()에서와 사용한 상수와 같음
 - SEEK_SET, SEEK_CUR, SEEK_END
- rewind()
 - 현재 파일의 오프셋을 처음으로 이동

ftello, fseeko

35

```
#include <stdio.h>

off_t ftello (FILE *fp );

int fseeko (FILE *fp, off_t offset, int whence );
```

- ftello
 - 성공시 현재 파일 위치 지시자, 오류시 -1
- fseeko
 - 성공시 0, 오류시 0이 아닌값
- ftell, fseek 와 비슷하고
반환값 타입이 off_t (32비트이상) 라는 것이 다름

fgetpos, fsetpos

36

```
#include <stdio.h>

int fgetpos (FILE *fp, fpos_t *pos );
int fsetpos (FILE *fp, const fpos_t *pos );
```

- fgetpos
 - 현재 파일 오프셋을 조회함
- fsetpos
 - 현재 파일 오프셋을 설정함
- ISO C 표준이 도입함

실습11:fgetpos, fsetpos 예제

37

- fgetpos와 fsetpos 사용하기
- 파일명 : fgetpos.c

```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
    FILE* fp;
    fpos_t pos;
    if((fp = fopen("./data","r")) == NULL){
        fprintf(stderr, "fopen error\n");
        exit(1);
    }
    pos.__pos = 10;
    if(fsetpos(fp, &pos) != 0)
        fprintf(stderr, "fsetpos\n");
    if(fgetpos(fp, &pos) != 0)
        fprintf(stderr, "fgetpos\n");
    printf("pos:%ld\n", pos.__pos);
    printf("sizeof:%ld\n", sizeof(pos.__pos));
    fclose(fp);
    return 0;
}
```

```
$ touch data
$ ./fgetpos
pos:10
sizeof:8
$
```

printf, fprintf, sprintf

38

```
#include <stdio.h>

int printf (const char *format,...);

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

int sprintf (char *buf ,const char *format,...);

int snprintf (char *buf, size_t n, const char *format, ...);
```

- printf
 - 표준 출력에 서식화된 문자들을 출력
- fprintf
 - 스트림에 서식화된 문자들을 출력
- sprintf
 - 메모리 공간에 서식화된 문자들을 출력
- snprintf
 - sprintf와 동일한 기능을 하고 버퍼의 크기를 명시적으로 지정가능

실습13:printf 예제

39

- printf의 출력예
- 파일명 : printf.c

```
#include<stdio.h>
int main()
{
    int x=10;
    int r;
    int max;
    char *string="hello, world";
    r=printf("x = %d\n",x);
    printf("r returned by printf() = %d\n\n",r);
    printf(":%s:           :%s:\n",string);
    printf(":%10s:          :%10s:\n",string);
    printf(":%15s:           :%15s:\n",string);
    printf(":%.10s:           :%.10s:\n",string);
    printf(":%-10s:           :%-10s:\n",string);
    printf(":%.15s:           :%.15s:\n",string);
    printf(":%15.10s:          :%15.10s:\n",string);
    printf(":%-15.10s:         :%-15.10s:\n\n",string);
    printf(":%.5d:             :%.5d:\n",123456789);
    printf(":%.5d:             :%.5d:\n",123);
    printf(":%.10d:            :%.10d:\n",123);
    printf(":%.5f:             :%.5f:\n\n", (double)123);
```

실습13:printf 예제

40

```

printf("주소표현\n");
printf("%%#.8x      %#.8x\n", (unsigned)123);
printf("0x%%08x     0x%08x\n", (unsigned)123);
printf("%%#10x      %#010x\n\n", (unsigned)123);
printf("시간표현\n");
printf("%%.2d:%%.2d:%%.2d      %.2d:%.2d:%.2d\n", 1, 59, 59);
printf("%%.2d:%%.2d:%%.2d      %.2d:%.2d:%.2d\n", 1, 1, 1);
printf("%%.2d:%%.2d:%%.2d      %.2d:%.2d:%.2d\n", 1, 10, 100);
printf("%%02d:%%02d:%%02d      %02d:%02d:%02d\n", 1, 59, 59);
printf("%%02d:%%02d:%%02d      %02d:%02d:%02d\n", 1, 1, 1);
printf("%%02d:%%02d:%%02d      %02d:%02d:%02d\n", 1, 10, 100);
printf("\n");
printf(":%%-d:      :%-d:\n", x);
printf(":%+d:      :%+d:\n", x);
printf(":% d:      :% d:\n", x);
printf(":%#x:      :%#x:\n", x);
printf(":%#o:      :%#o:\n", x);
printf(":%010d:      :%010d:\n", x);
printf("\n");
max=10;
printf("%%*. *s      :%*. *s:\n", 15, max, string);
return 0;
}

```


실습13:printf 예제

41

```
$ ./printf
x = 10
r returned by printf() = 7

:%s:           :hello, world:
:%10s:          :hello, world:
:%15s:           :  hello, world:
:%.10s:          :hello, wor:
:%-10s:          :hello, world:
:%.15s:          :hello, world:
:%15.10s:        :  hello, wor:
:%-15.10s:       :hello, wor   :

:%.5d:           :123456789:
:%.5d:           :00123:
:%.10d:           :0000000123:
:%.5f:           :123.00000:
```

주소표현

```
%#.8x           0x0000007b
0x%08x           0x0000007b
%#10x            0x0000007b
```

시간표현

```
%.2d:%.2d:%.2d      01:59:59
%.2d:%.2d:%.2d      01:01:01
%.2d:%.2d:%.2d      01:10:100
%02d:%02d:%02d      01:59:59
%02d:%02d:%02d      01:01:01
%02d:%02d:%02d      01:10:100
```

```
:%-d:           :10:
:%+d:           :+10:
:% d:           : 10:
:%#x:           :0xa:
:%#o:           :012:
:%010d:         :0000000010:

%*.*s           :  hello, wor:
$
```

실습12:fprintf, snprintf 예제

42

- formatted **입출력**
- **파일명** : fprintf.c

```
#include<stdio.h>

int main(void)
{
    char szTest[1024] = {0};
    fprintf(stdout,"%s : %d\n", "stdout", 777);
    fprintf(stderr,"%s : %d\n", "stderr", 777);
    snprintf(szTest, sizeof(szTest), "I love %s.\n", "you");
    printf("%s", szTest);
    snprintf(szTest, sizeof(szTest), "%s:%d\n", __FILE__, __LINE__);
    printf("%s", szTest);
    return 0;
}
```

```
$ ./fprintf > stdoutfile 2> stderrfile
$ cat stdoutfile
stdout : 777
I love you.
fprintf.c:10
$ cat stderrfile
stderr : 777
$
```

scanf, fscanf, sscanf

43

```
#include <stdio.h>

int scanf (const char *format,...);

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

int sscanf (const char *buf ,const char *format,...);
```

- **리턴 값:** 성공하면 읽은 문자 수, 실패하면 EOF
- **scanf()**
 - 표준 입력에서 서식화된 문자들을 입력
- **fscanf()**
 - 스트림에서 서식화된 문자들을 입력
- **sscanf()**
 - 메모리 버퍼에서 서식화된 문자들을 입력

실습14:scanf, fscanf, sscanf 예제

44

- scanf
- 파일명 : scanf.c

```
#include<stdio.h>

int main(void)
{
    FILE *fp;
    char szName[] = "advc";
    char szName2[1024];
    char szBuf[1024];
    int no = 10, no2;

    if((fp = fopen("./test", "w+")) == NULL){
        fprintf(stderr, "fopen error\n");
        return 1;
    }
    fprintf(fp, "name:%s no:%d\n", szName, no);
    rewind(fp);
    fscanf(fp, "name:%s no:%d\n", szName2, &no2);
    fclose(fp);
    fprintf(stdout, "name:%s no:%d\n", szName2, no2);

    strcpy(szBuf, "name:sanghun no:1101144127");
    sscanf(szBuf, "name:%s no:%d", szName2, &no2);
    fprintf(stdout, "name:%s no:%d\n", szName2, no2);
    return 0;
}
```

```
$ ./scanf
name:advc no:10
name:sanghun no:1101144127
$ cat ./test
name:advc no:10
$
```

실습15-1

45

- 파일명 : fprintf.c

```
#include<stdio.h>

int main(void)
{
    FILE *fp;
    int pin;
    int on;
    int off;
    int repeat;

    if((fp = fopen("./gpio_data2", "w")) == NULL){
        fprintf(stderr, "fopen error\n");
        return 1;
    }
    printf("gpio pin number:"); scanf("%d", &pin);
    printf("on period(sec):");   scanf("%d", &on);
    printf("off period(sec):");  scanf("%d", &off);
    printf("repeat:");          scanf("%d", &repeat);

    fprintf(fp, "pin:%d\n", pin);
    fprintf(fp, "on:%d\n", on);
    fprintf(fp, "off:%d\n", off);
    fprintf(fp, "repeat:%d\n", repeat);
    fclose(fp);

    return 0;
}
```

```
pi@robotcode ~ $ ./fprintf
gpio pin number:12
on period(sec):1
off period(sec):1
repeat:10
pi@robotcode ~ $ cat gpio_data2
pin:12
on:1
off:1
repeat:10
pi@robotcode ~ $
```

실습15-2

46

- 파일명 : fscanf.c

```
#include<stdio.h>

int main(void)
{
    FILE *fp;
    int pin;
    int on;
    int off;
    int repeat;

    if((fp = fopen("./gpio_data2", "r")) == NULL){
        fprintf(stderr, "fopen error\n");
        return 1;
    }
    fscanf(fp, "pin:%d\n", &pin);
    fscanf(fp, "on:%d\n", &on);
    fscanf(fp, "off:%d\n", &off);
    fscanf(fp, "repeat:%d\n", &repeat);

    printf("gpio pin number:%d\n", pin);
    printf("on period(sec):%d\n", on);
    printf("off period(sec):%d\n", off);
    printf("repeat:%d\n", repeat);
    fclose(fp);

    return 0;
}
```

```
pi@robotcode ~ $ ./fscanf
gpio pin number:12
on period(sec):1
off period(sec):1
repeat:10
pi@robotcode ~ $
```

tmpnam, tmpfile, tempnam

47

```
#include <stdio.h>

char *tmpnam (char *ptr);

FILE *tmpfile (void );

char *tempnam (const char *directory ,const char *prefix );
```

- **기능: 프로그램이 수행되는 동안만 존재하는 임시 파일을 만들거나, 임시 파일 이름을 만들 때**
- **tmpnam()**
 - 시스템에서 유일한 파일이름을 생성, 최대 TMP_MAX번 호출 가능
 - *ptr*
 - NULL이 아니면 L_tmpnam 만큼의 공간으로 파일 이름을 저장
 - NULL로 지정되면 정적영역에 저장됨
- **tmpfile()**
 - wb+ 모드로 파일을 생성하고, 스트림에 대한 포인터를 돌려 줌
- **tempnam()**
 - 생성하고자 하는 파일의 경로와 접두어(5자 이하)를 지정함

실습:tmpnam

48

- 임시파일 사용하기
- 파일명 : tmpnam.c

```
#include<stdio.h>

int main(void){
    char    name[L_tmpnam], line[1024];
    FILE    *fp;
    printf("%s\n", tmpnam(NULL));

    tmpnam(name);
    printf("%s\n", name);

    if ((fp = tmpfile()) == NULL){
        fprintf(stderr, "tmpfile error");
        return 1;
    }
    fputs("one line of output\n", fp);
    rewind(fp);
    if (fgets(line, sizeof(line), fp) == NULL){
        fprintf(stderr, "fgets error");
    }
    fputs(line, stdout);
    return 0;
}
```

```
$ ./tmpnam
/tmp/fileauFKtU
/tmp/fileUeVNZQ
one line of output
$
```


실습:tempnam 예제

49

- "a+" 파일스트림 열기
- 파일명 : tempnam.c

```
#include<stdio.h>

int main(int argc, char *argv[])
{
    if(argc != 3){
        fprintf(stderr, "usage: a.out <directory> <prefix>\n");
        return 1;
    }
    printf("%s\n", tempnam(argv[1][0] != ' ' ? argv[1] : NULL,
argv[2][0] != ' ' ? argv[2] : NULL));
    return 0;
}
```

```
$ ./tempnam /home temp
/home/tempvcz4tw
$
```

미션

50

- **실습 15 프로그램은 gpio_data2 파일을 읽고 내용을 출력한다.
gpio_data2 파일을 읽고 LED를 제어하도록 프로그램을 수정하시오.
(wiringPi 라이브러리 사용)**
 - pin : GPIO 핀번호
 - on : GPIO 핀이 On 상태인 시간(초)
 - off : GPIO 핀이 Off 상태인 시간(초)
 - repeat : 반복 회수