Chapter 7
Input & Output
입력과 출력

CSE2018 시스템프로그래밍기초
2016년 2학기

한양대학교 ERICA
컴퓨터공학과 => 소프트웨어학부
도경구

1. Standard Input and Output
2. Formatted Output - Printf
3. Variable-length Argument Lists
4. Formatted Input - Scanf
5. File Access
6. Error Handling - Stderr and Exit
7. Line Input and Output
8. Miscellaneous Functions

Standard Input

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

- Text stream = a sequence of lines
- Each lines ends with a newline character (\n)

`int getchar(void)`	호출할때마다 다음 입력 문자 하나를 내줌. 파일의 끝을 만나면 EOF(-1)를 내줌.

Input from file

`prog < infile`	prog가 입력을 키보드 대신 infile 파일에서 받음

Pipelining

`prog1 | prog2`	prog1과 prog2를 실행시키는데 prog1의 표준출력을 prog2의 표준입력으로 전달함

Standard Output

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

- Text stream = a sequence of lines
- Each lines ends with a newline character (\n)

`int putchar(int)`	putchar(c)는 문자 c를 표춘출력으로 내놓음 특별한 언급이 없으면 (by default) 출력창 결과로 그 문자를 내주는데, 도중에 오류가 발생하면 EOF를 내줌

Output to file

`prog > outfile`	prog가 출력을 출력창 대신 outfile 파일에 씀

Pipelining

`prog1 | prog2`	prog1과 prog2를 실행시키는데 prog1의 표준출력을 prog2의 표준입력으로 전달함

Standard Input & Output

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

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

/* lower: convert input to lower case */
int main() {
    int c;

    while ((c = getchar()) != EOF)
        putchar(tolower(c));
    return 0;
}
```

Formatted Output

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

`int printf (char *format, arg1, arg2, …)`	인수들을 format에 따라서 문자로 변환하고, 형태를 갖추어 표준출력에 프린트함. 결과로 프린트한 문자의 개수를 내줌
`int sprintf (char *string, char *format, arg1, arg2, …)`	printf와 기능은 동일하나, 출력을 첫 인수 문자열에 저장함

%

CHARACTER	ARGUMENT TYPE; PRINTED AS
d, i	int; decimal number.
o	int; unsigned octal number (without a leading zero).
x, X	int; unsigned hexadecimal number (without a leading 0x or 0X), using abcdef or ABCDEF for 10, ..., 15.
u	int; unsigned decimal number.
c	int; single character.
s	char *; print characters from the string until a '\0' or the number of characters given by the precision.
f	double; [−]*m.dddddd*, where the number of *d*'s is given by the precision (default 6).
e, E	double; [−]*m.ddddddx*±*xx* or [−]*m.ddddddX*±*xx*, where the number of *d*'s is given by the precision (default 6).
g, G	double; use %e or %E if the exponent is less than −4 or greater than or equal to the precision; otherwise use %f. Trailing zeros and a trailing decimal point are not printed.
p	void *; pointer (implementation-dependent representation).
%	no argument is converted; print a %.

Formatted Input

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

`int scanf (char *format, arg1, arg2, …)`	표준입력에서 문자들을 읽어, format 대로 해석한 다음, 결과를 인수에 저장함 프린트함. 인수는 반드시 포인터라야 함
`int sscanf (char *string, char *format, arg1, arg2, …)`	표준입력에서 읽지 않고 문자열에서 읽음

CHARACTER	INPUT DATA; ARGUMENT TYPE
`d`	decimal integer; `int *`.
`i`	integer; `int *`. The integer may be in octal (leading `0`) or hexadecimal (leading `0x` or `0X`).
`o`	octal integer (with or without leading zero); `int *`.
`u`	unsigned decimal integer; `unsigned int *`.
`x`	hexadecimal integer (with or without leading `0x` or `0X`); `int *`.
`c`	characters; `char *`. The next input characters (default 1) are placed at the indicated spot. The normal skip over white space is suppressed; to read the next non-white space character, use `%1s`.
`s`	character string (not quoted); `char *`, pointing to an array of characters large enough for the string and a terminating `'\0'` that will be added.
`e, f, g`	floating-point number with optional sign, optional decimal point and optional exponent; `float *`.
`%`	literal `%`; no assignment is made.

Formatted Input

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

```
#include <stdio.h>

/* rudimentary calculator */
int main() {
    double sum, v;

    sum = 0;
    while (scanf("%1f", &v) == 1)
        printf("\t%.2f\n", sum += v);
    return 0;
}
```

```
int day, year;
char monthname[20];

scanf("%d %s %d", &day, monthname, &year);
```

```
25 Dec 2016
```

```
int day, month, year;

scanf("%d/%d/%d", &month, &day, &year);
```

```
12/25/16
```

- format string 에서 공백은 무시
- 읽을 때도 공백은 무시

Formatted Input

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

```
int day, year;
char monthname[20];

scanf("%d %s %d", &day, monthname, &year);
```

25 Dec 2016

```
int day, month, year;

scanf("%d/%d/%d", &month, &day, &year);
```

12/25/16

- format string 에서 공백은 무시
- 읽을 때도 공백은 무시

```
while (getline(line, sizeof(line)) >0) {
    if (scanf(line, "%d %s %d", &day, monthname, &year) == 3)
        printf("valid: %s\n", line);    /* 25 Dec 2016 */
    else if (sscanf(line, "%d/%d/%d", &month, &day, &year) == 3)
        printf("valid: %s\n", line);    /* 12/25/16 */
    else
        printf("invalid: %s\n", line);
}
```

File Access

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

파일 열기

- 프로그램에서 파일을 읽거나 쓰기 전에 반드시 열어야 한다.

```
FILE *fp;
FILE *fopen(char *name, char *mode);
```

```
fp = fopen(name, mode)
```

파일이름 용도

"r"	read	읽기
"w"	write	쓰기
"a"	append	이어쓰기

- 쓰기 또는 이어쓰기 용도로 지정한 파일이 존재하지 않으면, 그 이름으로 하나 새로 만든다.
- 기존 파일에 쓰면, 기존의 내용은 모두 지워진다.
- 기존 파일에 이어쓰면, 기존의 내용을 그대로 두고 이어서 쓴다.

- 존재하지 않거나 접근권한이 없는 파일을 읽으려 하면 오류이고, NULL을 내준다.

File Access

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

파일 사용

`int getc(FILE *fp)`	파일에서 다음 문자를 가져와 내줌. 파일의 끝에 도달하거나 오류가 발생하면 EOF를 내줌
`int putc(int c, FILE *fp)`	문자 c를 파일 fp에 쓰고, 쓴 문자를 내줌 오류가 발생하면 EOF를 내줌

자동 오픈 파일	
stdin	키보드에 연결
stdout	출력창에 연결
stderr	출력창에 연결

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

파일 입출력
`int fscanf(FILE *fp, char *format, . . .)`
`int fprintf(FILE *fp, char *format, . . .)`

```c
#include <stdio.h>

/* cat: concatenate files, version 1 */
main(int argc, char *argv[]) {
    FILE *fp;
    void filecopy(FILE *, FILE *);

    if (argc == 1)
        filecopy(stdin, stdout);
    else
        while (--argc > 0)
            if ((fp = fopen(*++argv, "r")) == NULL) {
                printf("cat: can't open %s\n", *argv);
                return 1;
            }
            else {
                filecopy(fp, stdout);
                fclose(fp);
            }
    return 0;
}
```

```c
/* filecopy: copy file ifp to file ofp */
void filecopy(FILE *ifp, FILE *ofp) {
    int c;

    while ((c = getc(ifp)) != EOF)
        putc(c, ofp);
}
```

Error Handling

exit(n)	◦ 호출하면 프로그램 실행을 종료 ◦ n=0 이면, 정상종료; n>0 이면, 비정상종료 ◦ fclose를 자동 호출하여 열려있는 출력 파일을 모두 닫음 ◦ main 함수의 return expr 는 exit(expr)와 동일
int ferror (FILE *fp)	◦ fp 처리 도중 오류가 발생하면 0이 아닌 값을 내줌 ◦ 출력 오류가 그리 많이 발생하지 않긴 하지만, 공간이 꽉 찬 경우 발생하므로 반드시 호출하여 검사해야 함
int feof (FILE *fp)	◦ ferror와 유사하게 작동 ◦ 파일 끝을 발견하면 0이 아닌 값을 내줌

Error Handling

cat v.1의 문제점

- 파일에 접근할 수 없는 사유로 오류가 발생하면, 오류 메시지가 대신 붙게 된다.

해결책

- stdout 외에 별도의 출력 스트림 stderr을 사용한다.

```
/* cat: concatenate files, version 2 */
main(int argc, char *argv[]) {
    FILE *fp;
    void filecopy(FILE *, FILE *);
    char *prog = argv[0]; /* program name for errors */

    if (argc == 1)
        filecopy(stdin, stdout);
    else
        while (--argc > 0)
            if ((fp = fopen(*++argv, "r")) == NULL) {
                fprintf(stderr, "%s: can't open %s\n", prog, *argv);
                exit(1);
            }
            else {
                filecopy(fp, stdout);
                fclose(fp);
            }
    if (ferror(stdout)) {
        fprintf(stderr, "%s: error writing stdout\n", prog);
        exit(2);
    }
    exit(0);
}
```

Line Input & Output

char *fgets (char *line, int maxline, FILE *fp)	• fp에서 다음 입력 한 줄을 읽어 (newline 포함) 문자열 line에 저장 • maxline-1 개 문자까지 읽을 수 있음 • line의 맨 뒤에 '\0' 문자를 붙임 • 보통은 line을 내주는데, 파일의 끝에 도달하거나 오류가 발생하면 NULL을 내줌
int fputs (char *line, FILE *fp)	• 파일 fp에 문자열 line을 씀 • 오류가 발생하면 EOF를 내주고, 정상적인 경우 0을 내줌
gets	• fgets와 유사하게 작동 • 차이점은 stdin에서 작동함 • 맨끝의 '\n'을 지움
puts	• fputs와 유사하게 작동 • 차이점은 stdout에서 작동함 • 맨끝에 '\n'을 붙임

Line Input & Output

```
/* fgets: get at most n chars from iop */
char *fgets(char *s, int n, FILE *iop) {
    register int c;
    register char *cs;

    cs = s;
    while (--n > 0 && (c = getc(iop)) != EOF)
        if ((*cs++ = c) == '\n')
            break;
    *cs = '\0';
    return (c == EOF && cs == s) ? NULL : s;
}
```

```
/* fputs: put string s on file iop */
char *fputs(char *s, FILE *iop) {
    int c;

    while (c = *s++)
        putc(c, iop);
    return ferror(iop) ? EOF : 0;
}
```

```
/* readline: read a line, return length */
int readline(char *line, int max) {
    if (fgets(line, max, stdin) == NULL)
        return 0;
    else
        return strlen(line);
}
```

Misc. Functions

String Operations

`<string.h>`

```
char *s, *t;
int c, n;
```

`strcat(s,t)`	concatenate `t` to end of `s`
`strncat(s,t,n)`	concatenate `n` characters of `t` to end of `s`
`strcmp(s,t)`	return negative, zero, or positive for `s < t`, `s == t`, or `s > t`
`strncmp(s,t,n)`	same as `strcmp` but only in first `n` characters
`strcpy(s,t)`	copy `t` to `s`
`strncpy(s,t,n)`	copy at most `n` characters of `t` to `s`
`strlen(s)`	return length of `s`
`strchr(s,c)`	return pointer to first `c` in `s`, or `NULL` if not present
`strrchr(s,c)`	return pointer to last `c` in `s`, or `NULL` if not present

Misc. Functions

Character Class Testing and Conversion

`<ctype.h>`

```
int c;
```

`isalpha(c)`	non-zero if `c` is alphabetic, 0 if not
`isupper(c)`	non-zero if `c` is upper case, 0 if not
`islower(c)`	non-zero if `c` is lower case, 0 if not
`isdigit(c)`	non-zero if `c` is digit, 0 if not
`isalnum(c)`	non-zero if `isalpha(c)` or `isdigit(c)`, 0 if not
`isspace(c)`	non-zero if `c` is blank, tab, newline, return, formfeed, vertical tab
`toupper(c)`	return `c` converted to upper case
`tolower(c)`	return `c` converted to lower case

Command Execution

```
system(char *s)
```

```
system("date")
```

Misc. Functions

Storage Management

`void *malloc (size_t n)`	◦ 실행중 메모리 블록을 분양받음 ◦ 값이 지정되어 있지 않은 공간 n 바이트 분양받고 포인터를 내줌 ◦ 분양 불가하면 NULL을 내줌 ◦ 0으로 초기 값을 지정함
`void *calloc (size_t n, size_t size)`	◦ 실행중 메모리 블록을 분양받음 ◦ size 바이트 공간 n개를 저장할 충분한 공간을 분양받고 포인터를 내줌 ◦ 분양 불가하면 NULL을 내줌 ◦ 0으로 초기 값을 지정함
`free(p)`	◦ p가 가리키는 공간을 해제함 (재사용 가능하게 분양 취소) ◦ 해제 후 사용하려 하면 오류임

```
int *ip;
ip = (int *) calloc(n, sizeof(int));


for (p = head; p != NULL; p = p->next)  /* wrong */
    free(p);

for (p = head; p != NULL; p = q) {   /* right */
    q = p->next;
    free(p);
```

Misc. Functions

Mathematical Functions

`<math.h>`

All functions return `double`

`sin(x)`	sine of x, x in radians
`cos(x)`	cosine of x, x in radians
`atan2(y,x)`	arctangent of y/x, in radians
`exp(x)`	exponential function e^x
`log(x)`	natural (base e) logarithm of x ($x>0$)
`log10(x)`	common (base 10) logarithm of x ($x>0$)
`pow(x,y)`	x^y
`sqrt(x)`	square root of x ($x \geqslant 0$)
`fabs(x)`	absolute value of x

Misc. Functions

Random Number Generation

<stdlib.h>

rand()	0~RAND_MAX 범위내에서 무작위로 정수 시퀀스를 만들어냄
srand (unsigned)	rand에 사용할 시드 설정

무작위로 0~1 범위의 실수 만들기

```
#define frand() ((double) rand() / (RAND_MAX+1.0))
```