문자와 문자열 함수 (1)

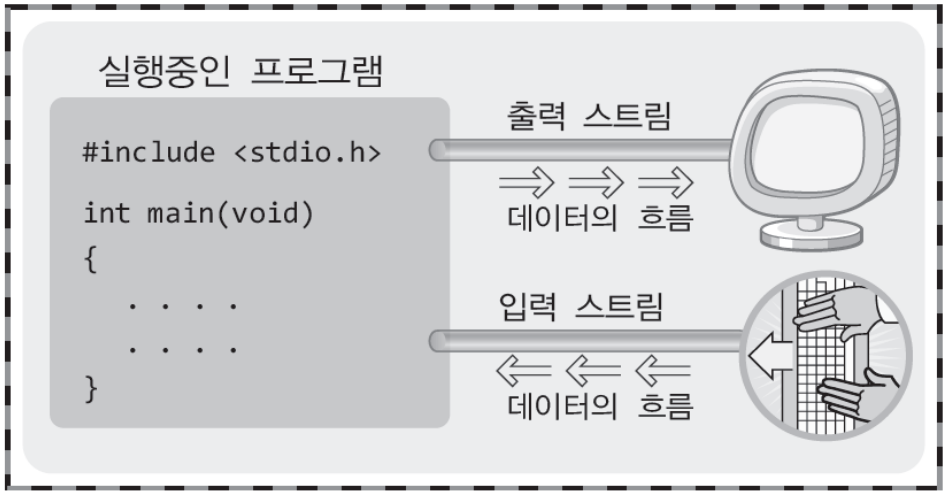
- 스트림, 문자/문자열 입출력 함수 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

스트림과 데이터의 이동

◎ 스트림 (stream)

- 한 방향(단 방향)으로 흐르는 데이터의 흐름
- 입력 스트림: 프로그램 안으로 데이터가 흘러오는 것
- 출력 스트림: 프로그램 밖으로 데이터가 흘러나가는 것



◎ 스트림의 생성과 소멸

- 콘솔 입출력을 위한 '입력 스트림'과 '출력 스트림'은 프로그램 실행 시 운영체제에 의해 자동으로 생성되고, 프로그램이 종료하면 자동으로 소멸됨 (표준 입출력 스트림)
- 파일 입출력을 위한 스트림은 개발자가 직접 운영체제에 요청을 해야함

스트림과 데이터의 이동

◎ 표준 입출력 스트림

• stdin	표준 입력 스트림	키보드 대상으로 입력
• stdout	표준 출력 스트림	모니터 대상으로 출력
• stderr	표준 에러 스트림	모니터 대상으로 출력

- stdout, stderr 모두 출력 대상이 모니터지만, 리다이렉션을 통해 데이터 전송 방향을 다르게 할 수 있음
- 표준 입출력 스트림은 프로그램 시작과 동시에 생성되어 프로그램 종료 시 자동으로 소멸
- 표준 입출력 스트림 외의 다른 스트림은 개발자가 직접 형성해야 함

문자 단위 입출력 함수

◎ 문자 출력 함수: putchar(), fputc()

```
#include <stdio.h>
int putchar(int c);
int fputc(int c, FILE * stream);
```

➜ 함수호출 성공 시 쓰여진 문자정보가, 실패 시 EOF 반환

- fputc() 함수는 문자를 전송할 스트림을 지정할 수 있으며, stdout을 전달하면 putchar() 함수와 동일하게 동작

◎ 문자 입력 함수: getchar(), fgetc()

```
#include <stdio.h>
int getchar(void);
int fgetc(FILE * stream);
```

➜ 파일의 끝에 도달하거나 함수호출 실패 시 EOF 반환

- fgetc() 함수는 문자를 입력 받을 스트림을 지정할 수 있으며, stdin을 전달하면 getchar() 함수와 동일하게 동작

문자 단위 입출력 함수

◎ ReadWriteChar.c

```
int main(void)
{
    int ch1, ch2;

    ch1=getchar();   // 문자 입력
    ch2=fgetc(stdin);   // 엔터 키 입력

    putchar(ch1);   // 문자 출력
    fputc(ch2, stdout);   // 엔터 키 출력
    return 0;
}
```

```
p
p
```

- 코드 상으로 두 개의 문자를 입출력 하고 있지만, 실제 Enter 키도 하나의 문자로 인식이 됨

문자 단위 입출력 함수

◎ 문자 입출력에서의 EOF

- EOF (End-of-File)
  - 파일의 끝을 표현하기 위해 정의해 놓은 상수
  - 파일을 대상으로 fgetc() 함수를 호출하면 파일의 끝에서 EOF(-1)가 반환
- 표준 입력(stdin)에서 fgetc(), getchar() 함수 호출로 EOF가 반환되는 경우
  - 명령 프롬프트(cmd)에서 ctrl+z 조합이 입력된 경우
  - 해당 함수 호출이 실패한 경우

문자 단위 입출력 함수

◎ ConsoleEOF.c

```
int main(void)
{
    int ch;
    while(1)
    {
        ch=getchar();
        if(ch==EOF)
            break;
        putchar(ch);
    }
    return 0;
}
```

```
Hi~
Hi~
I like C lang.
I like C lang.
^Z
```

- getchar(), fgetc() 함수의 반환값이 int 형인 이유?
  - char 형의 경우 unsigned char로 처리하는 컴파일러가 있기 때문에 -1을 인식할 수 있도록 int 형을 사용
- 입력된 데이터가 입력 스트림을 거쳐 프로그램으로 들어가는 시점은 Enter 키를 누른 시점

문자열 단위 입출력 함수

◎ 문자열 출력 함수: puts(), fputs()

```
#include <stdio.h>
int puts(const char * s);
int fputs(const char * s, FILE * stream);
```

➜ 성공 시 0이 아닌 값을, 실패 시 EOF 반환

- puts() 함수는 문자열 출력 후 자동으로 개행되지만, fputs() 함수는 자동으로 개행되지 않음

```
int main(void)
{
    char * str="Simple String";

    printf("1. puts test ------ \n");
    puts(str);
    puts("So Simple String");

    printf("2. fputs test ----- \n");
    fputs(str, stdout); printf("\n");
    fputs("So Simple String", stdout); printf("\n");

    printf("3. end of main ----\n");
    return 0;
}
```

```
1. puts test ------
Simple String
So Simple String
2. fputs test -----
Simple String
So Simple String
3. end of main ----
```

문자열 단위 입출력 함수

◎ 문자열 입력 함수: gets(), fgets()

```
#include <stdio.h>
char * gets(char * s);
char * fgets(char * s, int n, FILE * stream);
```

➜ 파일의 끝에 도달하거나 함수호출 실패 시 NULL 포인터 반환

```
int main(void)
{
    char str[7];   // 7바이트의 메모리 공간 할당
    gets(str);   // 입력 받은 문자열을 배열 str에 저장
    . . . .
}
```

- 배열의 길이를 넘어서는 길이의 문자열을 입력 받으면, 할당 받지 않은 메모리 공간을 침범하여 실행 중 오류 발생 가능

```
int main(void)
{
    char str[7];
    fgets(str, sizeof(str), stdin);
    . . . .  // stdin으로부터 문자열 입력 받아서 str에 저장
}
```

- stdin으로부터 문자열을 입력 받아 str에 저장하되, sizeof(str)의 길이 만큼만 저장해라

  "123456789" 입력 시, "123456"만 문자 배열에 저장됨 (맨 마지막 자리엔 널문자)

문자열 단위 입출력 함수

◎ ReadString.c

```
int main(void)
{
    char str[7];
    int i;

    for(i=0; i<3; i++)
    {
        fgets(str, sizeof(str), stdin);
        printf("Read %d: %s \n", i+1, str);
    }
    return 0;
}
```

```
12345678901234567890
Read 1: 123456
Read 2: 789012
Read 3: 345678
```

```
We
Read 1: We

like
Read 2: like

you
Read 3: you
```

```
Y & I
Read 1: Y & I

ha ha
Read 2: ha ha

^^ --
Read 3: ^^ --
```

- fgets() 함수는 Enter 키('₩n') 정보까지도 문자열의 일부로 저장하며, scanf() 함수와 다르게 공백을 포함한 문자열도 입력 받을 수 있음