문자와 문자열 함수 (2)

- 입출력 버퍼, 문자열 함수 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

표준 입출력과 버퍼

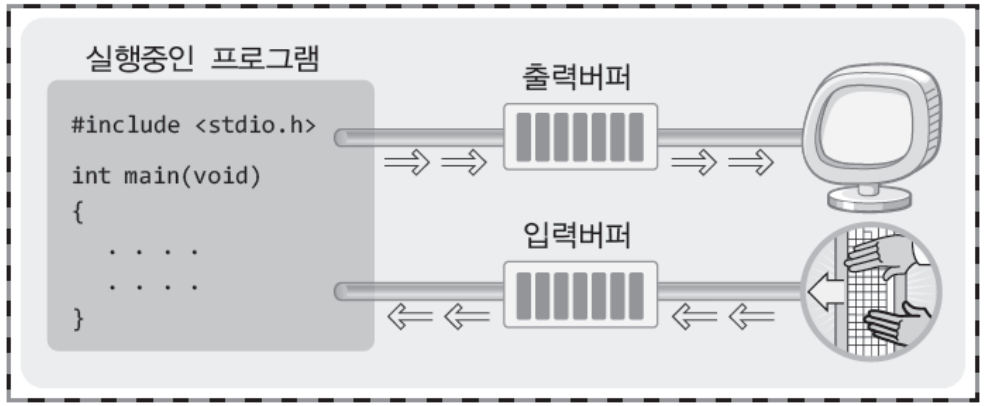
◎ 표준 입출력 함수 (standard I/O functions)

- ANSI C의 표준에서 정의한 입출력 함수
- 표준 입력 함수
  - scanf(), getchar(), fgetc(), gets(), fgets()
- 표준 출력 함수
  - printf(), putchar(), fputc(), puts(), fputs()
- 표준 입출력 함수를 이용하여 데이터를 입출력 하는 경우, 운영체제가 제공하는 '메모리 버퍼'를 통과함

표준 입출력과 버퍼

◎ 메모리 버퍼 (memory buffer)

- 데이터를 임시로 모아두는 메모리 공간
- 키보드로 입력된 데이터는 입력 버퍼에 저장된 후 프로그램에 의해 읽힘
- 입력된 데이터가 입력 스트림을 거쳐 입력 버퍼로 들어가는 시점은 엔터(Enter)키가 눌리는 시점
- 효율적인 데이터 전달을 위해 데이터 버퍼링(buffering)을 사용함



표준 입출력과 버퍼

◎ 입력 버퍼 비우기

- 입력 버퍼에 남아있는 불필요한 데이터를 소멸시키기 위해 필요

```
int main(void)
{
    char perID[7];
    char name[10];

    fputs("주민번호 앞 6자리 입력: ", stdout);
    fgets(perID, sizeof(perID), stdin);

    fputs("이름 입력: ", stdout);
    fgets(name, sizeof(name), stdin);

    printf("주민번호: %s \n", perID);
    printf("이름: %s \n", name);
    return 0;
}
```

실행결과1

```
주민번호 앞 6자리 입력: 950915
이름 입력: 주민번호: 950915
이름:
```

엔터 키가 남아서 문제가 되는 상황

실행결과2

```
주민번호 앞 6자리 입력: 950709-1122345
이름 입력: 주민번호: 950709
이름: -1122345
```

입력 버퍼에 6자리 넘은 문자열이 들어온 상황

```
void ClearLineFromReadBuffer(void)
{
    while(getchar()!='\n');
}

int main(void)
{
    char perID[7];
    char name[10];

    fputs("주민번호 앞 6자리 입력: ", stdout);
    fgets(perID, sizeof(perID), stdin);
    ClearLineFromReadBuffer();   // 입력버퍼 비우기

    fputs("이름 입력: ", stdout);
    fgets(name, sizeof(name), stdin);

    printf("주민번호: %s\n", perID);
    printf("이름: %s\n", name);
    return 0;
}
```

문자열 함수

◎ 문자열 길이를 반환하는 함수: strlen()

```
#include <string.h>
size_t strlen(const char * s);
```

➜ 전달된 문자열의 길이를 반환하되, 널 문자는 길이에 포함하지 않는다.

```
int main(void)
{
    char str[]="1234567";
    printf("%u \n", strlen(str));
    . . . .      // 문자열의 길이 7이 출력
}
```

- typedef unsigned int size_t; // unsigned int 형의 선언을 size_t로 대체함

문자열 함수

◎ RemoveBSN.c

- fgets()에 딸려오는 '₩n'을 제거하는 함수를 작성

```
void RemoveBSN(char str[])
{
    int len=strlen(str);
    str[len-1]=0;
}

int main(void)
{
    char str[100];
    printf("문자열 입력: ");
    fgets(str, sizeof(str), stdin);
    printf("길이: %d, 내용: %s \n", strlen(str), str);

    RemoveBSN(str);
    printf("길이: %d, 내용: %s \n", strlen(str), str);
    return 0;
}
```

```
문자열 입력: Good morning
길이: 13, 내용: Good morning

길이: 12, 내용: Good morning
```

문자열 함수

◎ 문자열을 복사하는 함수: strcpy(), strncpy()

```
#include <string.h>
char * strcpy(char * dest, const char * src);
char * strncpy(char * dest, const char * src, size_t n);
```

➜ 복사된 문자열의 주소 값 반환

- strcpy(dest, src) : src에 저장된 문자열을 dest에 복사
  - 문자열 src의 길이가 dest 배열의 길이보다 커, 배열의 범위를 넘어선 복사를 할 수 있음
- strncpy(dest, src, n) : src에 저장된 문자열 중에 크기 n 만큼만 dest에 복사

```
int main(void)
{
    char str1[30]="Simple String";
    char str2[30];
    strcpy(str2, str1);
    . . . .    // str1의 문자열을 str2에 복사
}
```

```
int main(void)
{
    char str1[30]="Simple String";
    char str2[30];
    strncpy(str2, str1, sizeof(str2));
    . . . .
}
```

문자열 함수

◎ StringCopyCase.c

- 문자열 복사 함수는 문자열 끝을 의미하는 널 문자('₩0')의 복사를 보장하지 않음

```
int main(void)
{
    char str1[20]="1234567890";
    char str2[20];
    char str3[5];

    /**** case 1 ****/
    strcpy(str2, str1);
    puts(str2);

    /**** case 2 ****/
    strncpy(str3, str1, sizeof(str3));
    puts(str3);

    /**** case 3 ****/
    strncpy(str3, str1, sizeof(str3)-1);
    str3[sizeof(str3)-1]=0;
    puts(str3);
    return 0;
}
```

```
1234567890
12345徼徼徼徼徼?234567890
1234
```

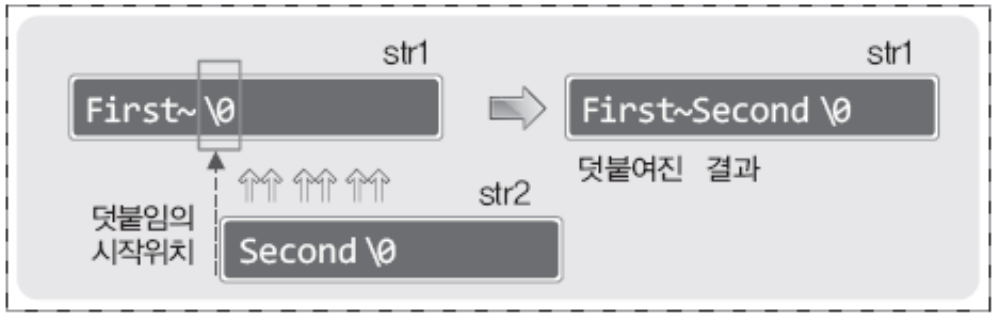
문자열 함수

◎ 문자열 덧붙임 함수 : strcat(), strncat()

```
#include <string.h>
char * strcat(char * dest, const char * src);
char * strncat(char * dest, const char * src, size_t n);
```

➜ 덧붙여진 문자열의 주소 값 반환

```
int main(void)
{
    char str1[30]="First~";
    char str2[30]="Second";
    strcat(str1, str2);
    . . . .  // str1의 문자열 뒤에 str2를 복사
}
```



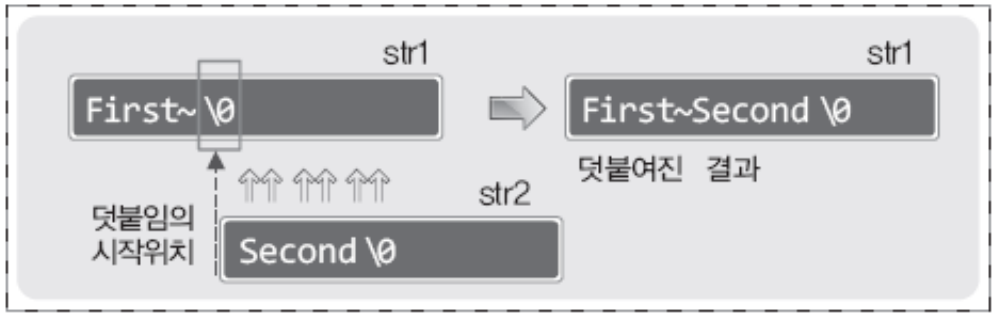
문자열 함수

◎ 문자열 덧붙임 함수 : strcat(), strncat()

```
#include <string.h>
char * strcat(char * dest, const char * src);
char * strncat(char * dest, const char * src, size_t n);
```

➜ 덧붙여진 문자열의 주소 값 반환

```
int main(void)
{
    char str1[30]="First~";
    char str2[30]="Second";
    strcat(str1, str2);
    . . . .  // str1의 문자열 뒤에 str2를 복사
}
```



- 문자열 덧붙임이 시작되는 위치는 널 문자 다음이 아닌, 널 문자가 저장된 위치부터임

문자열 함수

◎ StringConcatCase.c

```
int main(void)
{
    char str1[20]="First~";
    char str2[20]="Second";
    char str3[20]="Simple num: ";
    char str4[20]="1234567890";
    /**** case 1 ****/
    strcat(str1, str2);
    puts(str1);
    /**** case 2 ****/
    strncat(str3, str4, 7);
    puts(str3);
    return 0;
}
```

```
First~Second
Simple num: 1234567
```

- strncat(str1, str2, n)은 문자열 str1에 문자열 str2 중에 최대 n개를 덧붙이라는 의미로 n개에는 널 문자가 포함되지 않음
- strncpy() 함수와는 다르게 문자열의 끝에 널 문자를 자동으로 삽입해줌

문자열 함수

◎ 문자열을 비교하는 함수: strcmp(), strncmp()

```
#include <string.h>
int strcmp(const char * s1, const char * s2);
int strncmp(const char * s1, const char * s2, size_t n);
```

➜ 두 문자열의 내용이 같으면 0, 같지 않으면 0이 아닌 값 반환

- ASCII 코드 값을 기준으로 문자열의 대소를 비교
- 문자열 s1이 크면 양수를 반환하고, 문자열 s2가 크면 음수를 반환함 (컴파일러에 따라 반환되는 값은 다름)
- 사전 편찬 순서를 기준으로 앞에 위치하는 문자열이 작은 문자열이고, 뒤에 위치하는 문자열이 큰 문자열

문자열 함수

◎ StringCompCase.c

```
int main(void)
{
    char str1[20];
    char str2[20];
    printf("문자열 입력 1: ");
    scanf("%s", str1);
    printf("문자열 입력 2: ");
    scanf("%s", str2);

    if(!strcmp(str1, str2))
    {
        puts("두 문자열은 완벽히 동일합니다.");
    }
    else
    {
        puts("두 문자열은 동일하지 않습니다.");

        if(!strncmp(str1, str2, 3))
            puts("그러나 앞 세 글자는 동일합니다.");
    }
    return 0;
}
```

```
문자열 입력 1: Simple
문자열 입력 2: Simon
두 문자열은 동일하지 않습니다.
그러나 앞 세 글자는 동일합니다.
```

문자열 함수

◎ 문자열 변환 함수: atoi(), atol(), atof()

- 헤더파일 <stdlib.h>에 정의되어 있으며, 문자열로 표현된 정수나 실수 값을 정수형, 실수형 데이터로 변환하여 반환

```
int atoi(const char * str);        문자열의 내용을 int형으로 변환
long atol(const char * str);       문자열의 내용을 long형으로 변환
double atof(const char * str);     문자열의 내용을 double형으로 변환
```

```
int main(void)
{
    char str[20];
    printf("정수 입력: ");
    scanf("%s", str);
    printf("%d \n", atoi(str));
    printf("실수 입력: ");
    scanf("%s", str);
    printf("%g \n", atof(str));
    return 0;
}
```

```
정수 입력: 15
15
실수 입력: 12.456
12.456
```