컴퓨터 프로그래밍 I
(CSE2003-3)

Mon/Wed 16:30-17:45 pm

Lecture 24

String (Continued)

String input/output function

- scanf와 printf 함수를 이용하여 문자열의 입출력이 가능하다.
- scanf() 를 이용한 문자열 입력

```
scanf("%s", month);
```

month가 입력 받을 데이터를 저장하기에 충분한 크기인지를 고려할 필요가 있다.

1) 충분히 큰 배열을 선언
2) malloc 함수를 쓴다.
(사용자로부터 문자열의 길이를 입력받아야 함)

- [경우1] 길이 9의 문자열을 입력 받아 배열에 저장

```
char month[10];
scanf("%9s", month);
```

null까지 저장을 해야 하므로 길이 10인 배열을 만든다.

9글자 이상의 문자를 입력 받더라도 최대 9글자까지만을 month에 저장

① 학번: 8글자
char student id [9]
② 이름 : 공백 X

Note

%s로 받을 경우, blank(‘ ’), tab(‘\t’)등을 입력 받을 수 없다.

String input/output function

- **scanf() 를 이용한 문자열 입력(cont.)**
  - [경우2] edit set을 이용한 매치되는 문자만 저장

```
scanf("%10[0123456789.,-$]", str);
```

%[…] 은 edit set이라 불리며 scanf()를 통해 전달 받은 문자열이 […]안의 문자와 매치되는 경우에만 str[] 변수에 저장된다.

    - "-430,743.33$"와 같은 문자열을 입력 받는다

  - [경우3] edit set과 반대의 기능을 이용하여 저장

```
scanf("%81[^\n]", str);
```

%[ ]안의 ^는 scanf()를 통해 전달 받은 문자열이 ^ 뒤에 나오는 문자와 매치되지 않는 경우에만 str[] 변수에 저장된다.

    - '₩n'를 입력 받기 전까지 모든 문자열을 입력 받는다.

Note

조건을 만족하지 못하는 곳과 그 뒷부분은 모두 무시당한다.

String input/output function

- scanf 함수를 이용한 문자열 입력의 예제 프로그램

```c
#include <stdio.h>

int main(void)
{
        char str[10];
        char *p;

        printf("Enter a string: ");
        scanf("%9s", str);
        p=str;

        while(*p != '\0') {
                printf("%c", *p);
                p++;
        }

        printf("\n");
}
```

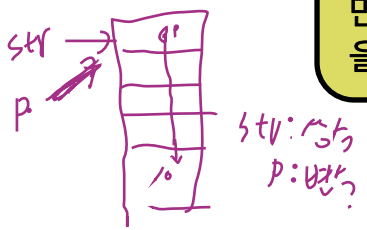
size

pointer variable

```
vore@nlpsag:~$ ./a.out
Enter a string: 123456789012345
123456789
vore@nlpsag:~$ ./a.out
Enter a string: 123 456789
123
vore@nlpsag:~$
```

최대 9글자를 입력 받는다.

%s로 받을 경우, Blank character ' '가 입력되면 그 앞까지만 받는 것을 확인할 수 있다.



(while 루프를 이용하여) 한 문자씩 출력한다.

String input/output function

- **printf() 를 이용한 문자열의 출력**
  - scanf에서와 같이 **%s**를 이용하여 문자열을 출력한다.
  - Null character('₩0')앞 까지만 출력하므로, 문자열의 중간에 null이 있으면 뒷부분은 출력되지 않는다.
  - 30칸에 맞춰서 출력. 오른쪽 정렬

```
printf("|%30s|\n", "This is the string");
```

```
|            This is the string|
```

  - 30칸에 맞춰서 출력. 왼쪽 정렬

```
printf("|%-30s|\n", "This is the string");
```

```
|This is the string            |
```

  - 15칸에 맞춰서 출력. 14글자만 출력. 왼쪽 정렬

```
printf("|%-15.14s|\n", "12345678901234567890");
```

```
|12345678901234 |
```

String input/output function

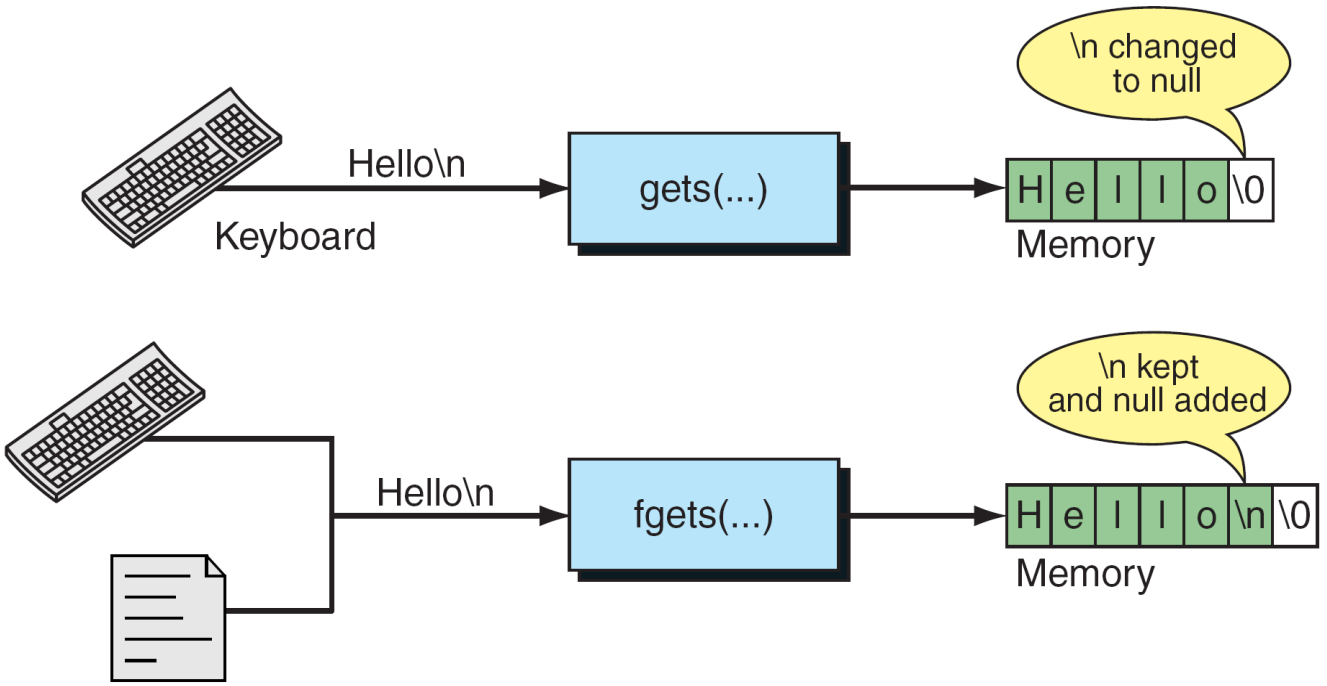
- printf를 이용한 string의 출력 예제 프로그램

```c
#include <stdio.h>

int main(void){

    printf("|%30s|\n", "This is the string");
    printf("|%-30s|\n", "This is the string");
    printf("|%-15.14s|\n", "123456789012345678");
    printf("|%15.14s|\n", "123456789012345678");

}
```

```
./a.out
|            This is the string|
|This is the string             |
|12345678901234 |
| 12345678901234|
vore@nlpsag:~$
```

String input/output function

- **gets / fgets**
  - format에 따라 입력을 받는 scanf, fscanf 등과 달리 formatting 없이 한 line을 읽어 들이는 함수.



String input/output function

- **gets / fgets 함수 proto-type**
  - char* gets(char* strPtr);

    성공 : strPtr의 주소를 return
    실패 : NULL

    - 키보드로부터 한 line을 입력받아 strPtr에 저장
    - 한 line은 '₩n'을 입력 받을 때까지를 의미, '₩n'은 '₩0'으로 치환되어 저장된다.
    - 한 line의 길이가 strPtr의 길이보다 길면 strPtr 뒷부분의 메모리가 침범되어 Segmentation fault가 발생하므로 주의해야 한다.

  - char* fgets(char* strPtr, int size, FILE *fp);

    성공 : strPtr의 주소를 return
    실패, 파일의 끝(EOF) : NULL

    - file 포인터 fp로부터 한 line을 읽어 들여 strPtr에 저장

    ➔ 읽어 들이는 문자의 최대 개수는 size-1개이다.

    - 한 line은 '₩n'을 입력 받을 때까지를 의미, '₩n'까지 입력 받고 뒤에 '₩0'이 추가로 저장됨.
    - File 포인터에 stdin을 주면 키보드로부터 입력을 받을 수 있다.

    ➔ 일반적으로 gets(strPtr)보다 fgets(strPtr, sizeof(strPtr), stdin); 의 사용이 권장된다.

String Input/Output Functions

- fgets 를 사용한 예제 프로그램

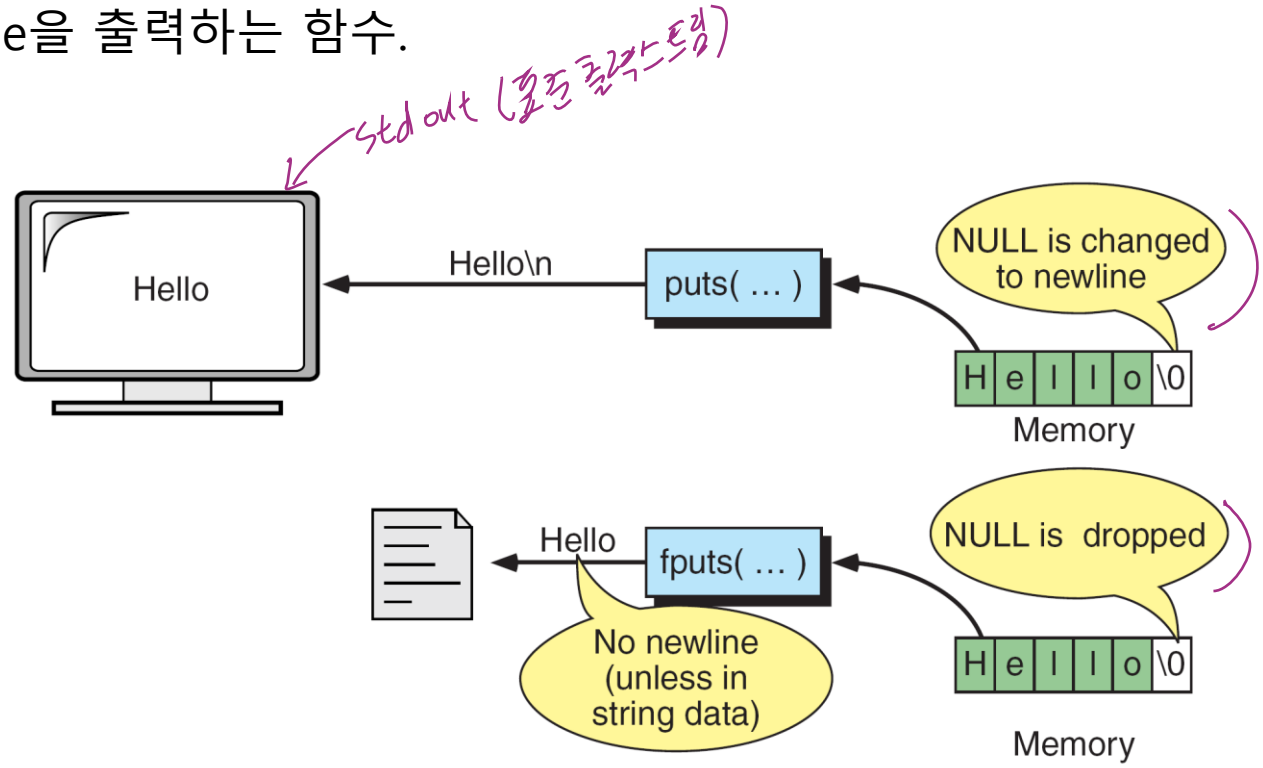
```c
#include <stdio.h>

int main(void)
{

        char str[81];

        printf("Please enter a string: ");
        fgets(str, sizeof(str), stdin);
        printf("Here is your string: \n\t%s", str);

        return 0;
}
```

키보드를 통해 한 줄을 입력받는다.

```
[root@mclab chap11]# ./chap11-3
Please enter a string: Now is the time for all students
Here is your string:
        Now is the time for all students
[root@mclab chap11]#
```

String input/output function

- **puts / fputs**
  - format에 따라 출력을 하는 printf, fprinf 등과 달리 formatting 없이 한 line을 출력하는 함수.



String input/output function

- **puts / fputs 함수 proto-type**
  - int puts (const char *strPtr);
    - strPtr의 문자열을 모니터에 한 line으로 출력
    - '₩0'이 '₩n'으로 치환되므로 문자열 끝에 '₩n'이 있다면 줄 바꿈이 두 번 일어난다.
    - '₩0'까지만 출력하므로 문자열 중간에 '₩0'이 있으면 뒷부분은 무시된다.

성공 : 출력한 글자 수 ('\n'로 치환된 '\0' 포함)
실패 : EOF(-1)

  - int fputs (const char *strPtr, FILE *fp);
    - strPtr의 문자열을 file 포인터 fp에 기록
    - strPtr은 '₩0'으로 끝나야 하며 NULL문자는 출력되지 않는다.
    - '₩0'까지만 출력하므로 문자열 중간에 '₩0'이 있으면 뒷부분은 무시된다.
    - File 포인터 fp에 stdout을 주면 모니터에 출력할 수 있다.

성공 : 1
실패 : EOF(-1)

➔ gets()와 fgets()의 큰 차이와 달리 puts(strPtr)와 fputs(strPtr, stdout)은 '₩n'이 마지막에 출력되는가의 여부, 함수의 return값만 다르다.

String Input/Output Functions

- fputs 를 사용한 예제 프로그램

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5
6          char str[]="Necessity Is the Mother of Invention.";
7          char* pStr = str;
8
9          fputs(pStr, stdout);
10         fputs("\n", stdout);
11         fputs(pStr+13, stdout);
12         fputs("\n", stdout);
13
14         return 0;
15 }
```

문자열의 시작위치를 지정

문자열의 시작부터 모두 출력

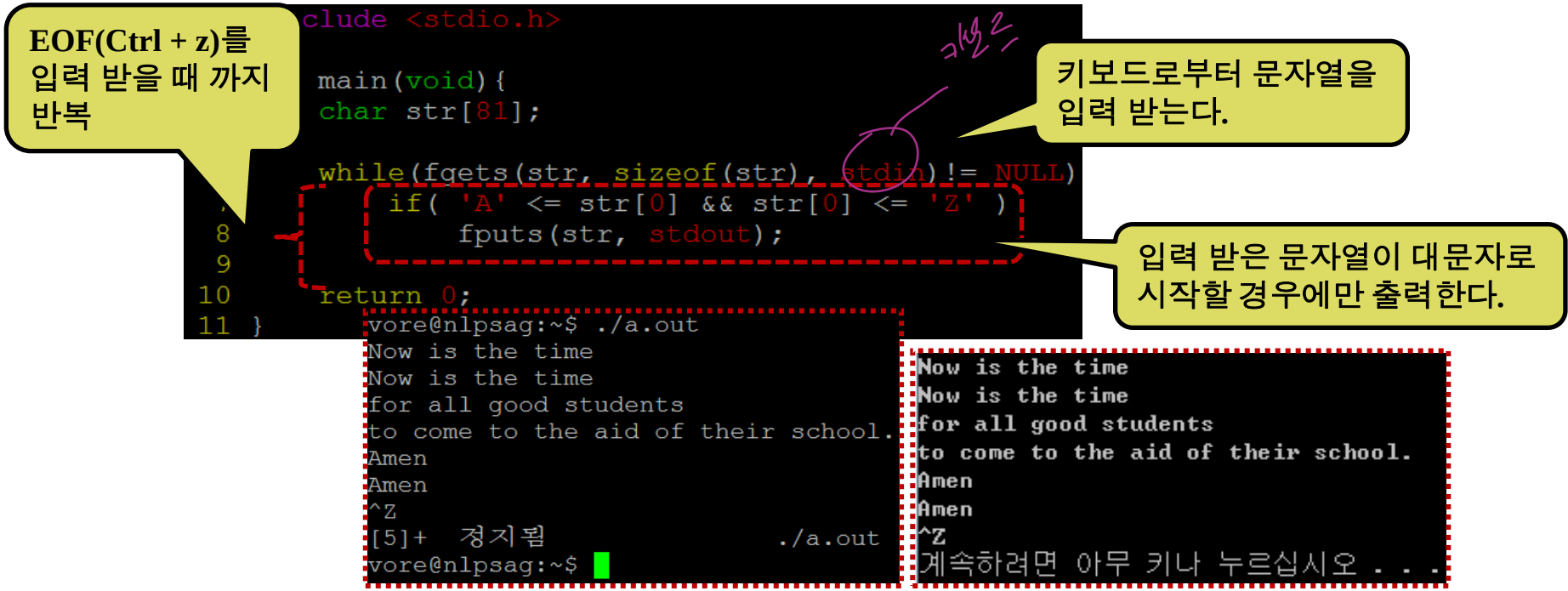
문자열의 13번째 문자부터 출력

```
[root@mclab chap11]# ./chap11-4
Necessity Is the Mother of Invention.
the Mother of Invention.
[root@mclab chap11]#
```

String Input/Output Functions

◆ fgets / fputs 사용 예제프로그램

- 다음 프로그램은 키보드로 입력 받아서 대문자로 시작할 경우에만 출력한다.



- Linux에서 ^Z(Ctrl+Z, EOF)를 입력하면 프로그램이 강제종료 되지만, Windows에서는 그렇지 않고 fgets에서 NULL이 return되어 while문을 빠져나가는 것을 볼 수 있다.

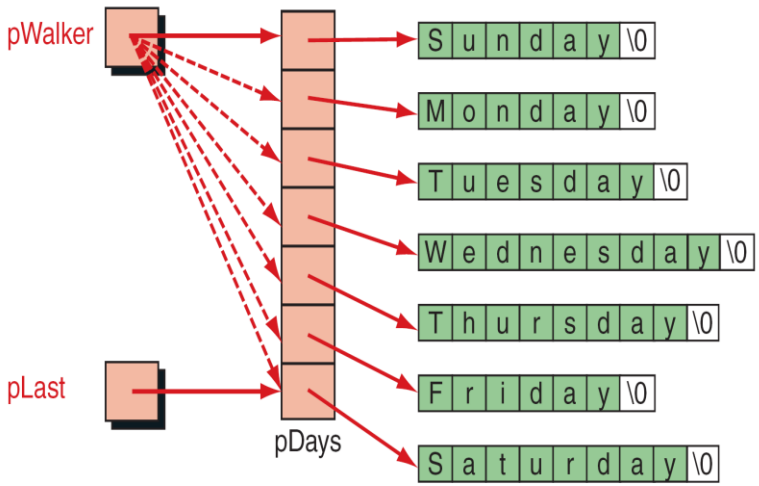
Arrays of strings

- **문자열을 원소로 갖는 배열을 만들어 사용할 수 있다.**
  - char *타입의 배열을 만들면 각각의 원소(포인터)가 문자열을 포인팅하도록 할 수 있다.
  - **예제 프로그램 - 문자열의 배열을 이용하여 일주일의 요일을 출력**

```c
#include <stdio.h>

int main(void)
{
        char* pDays[7];
        char** pLast;
        char** pWalker;

        pDays[0] = "Sunday";
        pDays[1] = "Monday";
        pDays[2] = "Tuesday";
        pDays[3] = "Wednesday";
        pDays[4] = "Thursday";
        pDays[5] = "Friday";
        pDays[6] = "Saturday";

        printf("The days of the week\n");
        pLast = pDays + 6;

        for(pWalker=pDays; pWalker <= pLast; pWalker++)
                printf("%s\n", *pWalker);

        return 0;
}
```

pDays에 각 문자열을 저장
문자열은 임의의 장소에 저장되며
pDays[0]등의 배열 원소는
그 문자열의 시작 주소를 갖는다.

pDays의 내용을 순차적으로 출력



Arrays of strings

- **문자열을 원소로 갖는 배열을 만들어 사용할 수 있다.**
  - char *타입의 배열을 만들면 각각의 원소(포인터)가 문자열을 포인팅하도록 할 수 있다.
  - **예제 프로그램 - 문자열의 배열을 이용하여 일주일의 요일을 출력**

```c
#include <stdio.h>

int main(void)
{
        char* pDays[7];
        char** pLast;
        char** pWalker;

        pDays[0] = "Sunday";
        pDays[1] = "Monday";
        pDays[2] = "Tuesday";
        pDays[3] = "Wednesday";
        pDays[4] = "Thursday";
        pDays[5] = "Friday";
        pDays[6] = "Saturday";

        printf("The days of the week\n");
        pLast = pDays + 6;

        for(pWalker=pDays; pWalker <= pLast; pWalker++)
                printf("%s\n", *pWalker);

        return 0;
}
```

pDays에 각 문자열을 저장
문자열은 임의의 장소에 저장되며
pDays[0]등의 배열 원소는
그 문자열의 시작 주소를 갖는다.

pDays의 내용을 순차적으로 출력

```
[root@mclab chap11]# ./chap11-6
The days of the week
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
[root@mclab chap11]#
```

String manipulation function

- **C에서는 문자열을 관리하는 여러 함수들을 제공한다.**

  #include <string.h>를 통해 사용할 수 있다

  - strlen() – 문자열의 길이 계산하는 함수
  - strcpy(), strncpy() – 문자열을 복사하는 함수
  - strcmp(), strncmp() – 두 문자열을 비교하는 함수
  - strcat(), strncat() – 두 문자열을 결합하는 함수
  - strtok() – 문자열을 자르는 함수

- **strlen()**

  - strlen은 string length의 약자로, 이 함수는 문자열의 길이를 계산해준다.
  - 문자열의 길이란 문자열의 시작 주소부터 NULL 문자 사이의 문자 개수를 의미한다.

```
int strlen (const char *string);
```

Return : *string부터 처음 만나는 '\0'까지의 길이

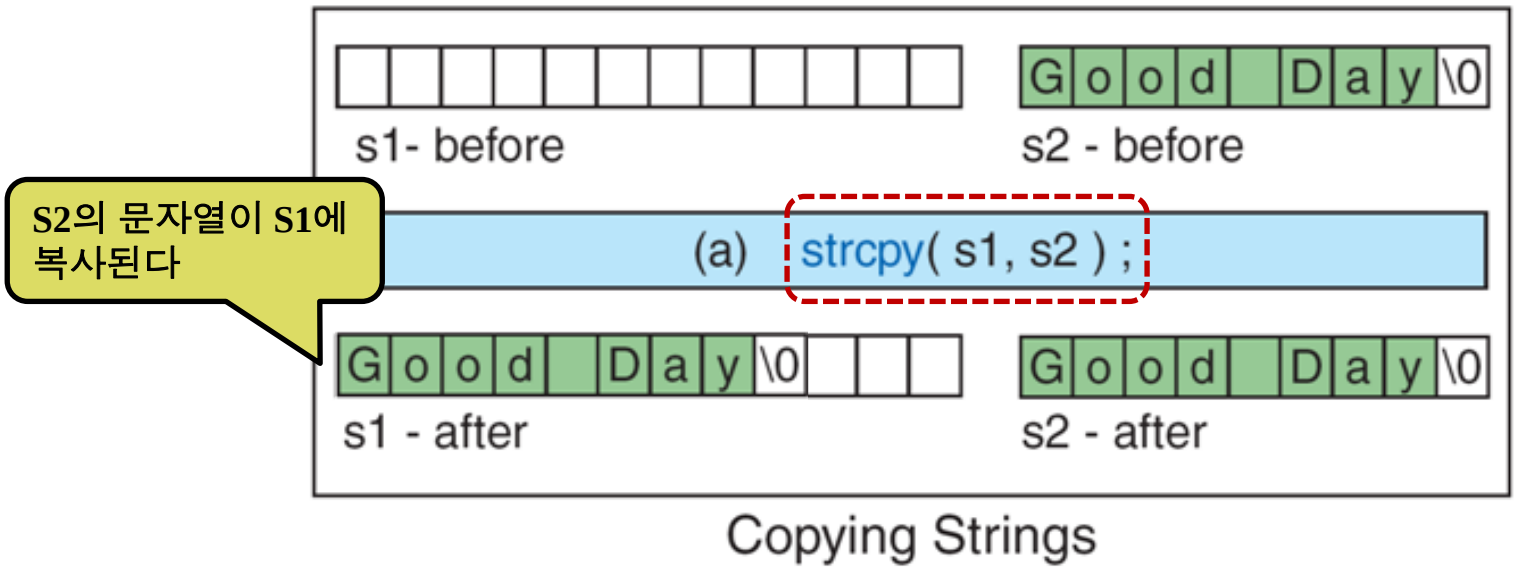
String manipulation function

- **strcpy()**
  - strcpy는 string copy의 약자로, 이 함수는 문자열을 복사하여 다른 배열 변수에 저장한다.

```
char *strcpy (char *to_strng, const char *from_strng);
```

Return : to_strng

  - from_string에 저장된 문자열을 to_string에 복사한다.



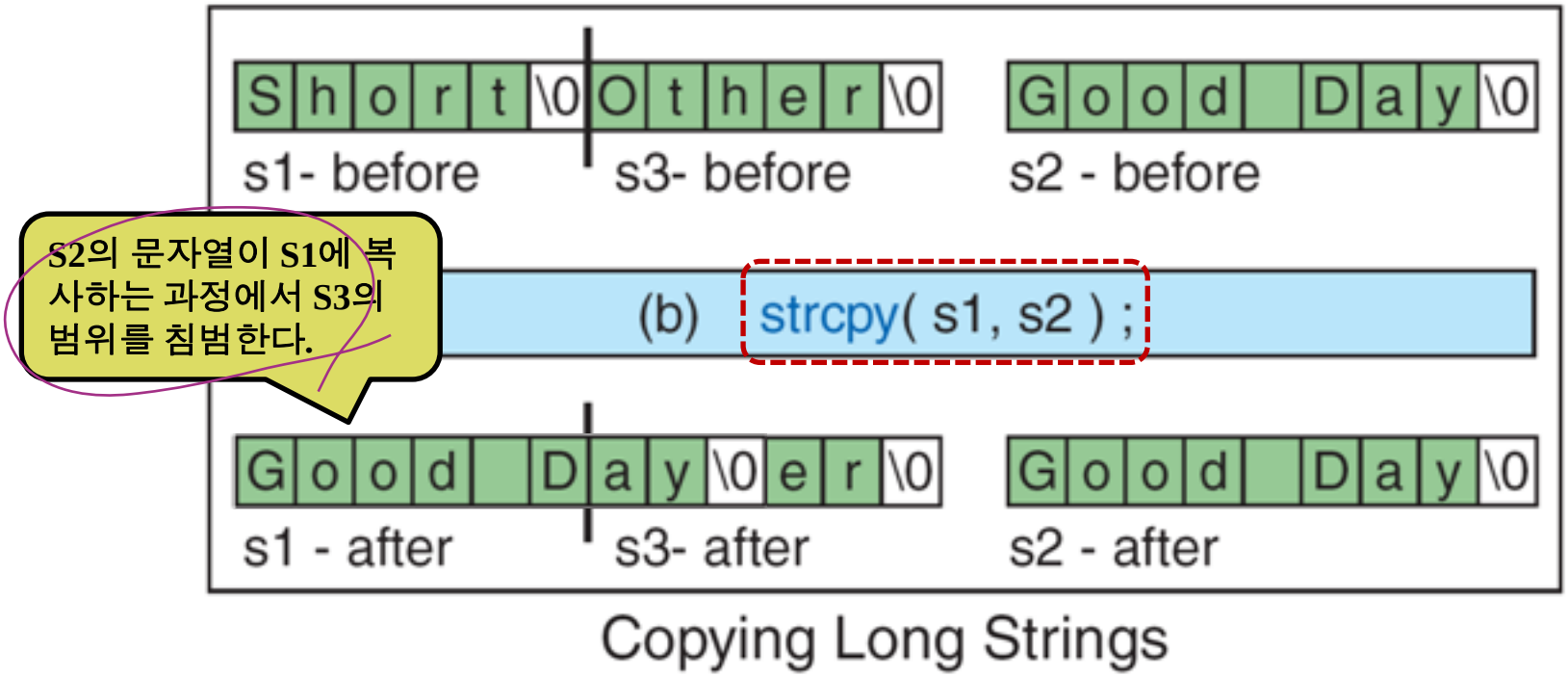
Copying Strings

String manipulation function

- **strcpy()**
  - char *sp;가 있을 때, sp에는 문자열 상수 시작 주소를 직접 대입할 수 있다.
    - sp = “I need openlab”;
  - 그러나 문자열 배열 str이 char str[20]; 으로 정적으로 선언되어 있다고 하면, str = “I need openlab”; 와 같은 표현은 불가능하다.
  - 따라서 문자 배열에 문자열을 대입하고자 할 때는 strcpy, memcpy등의 copy 함수를 이용하여 복사해주어야 한다.

String manipulation function

- **strcpy() 사용시 발생할 수 있는 문제**
  - to_string의 저장 공간이 from_string에 저장된 문자열의 크기보다 작을 경우
    - strcpy()는 to_string의 저장공간 이외의 인접 메모리 공간을 침범한다.



Copying Long Strings

String manipulation function

- **strncpy()**
  - strncpy는 strcpy처럼 문자열을 복사하는 함수이다.
  - strncpy 함수는 복사할 문자열의 크기를 지정할 수 있다

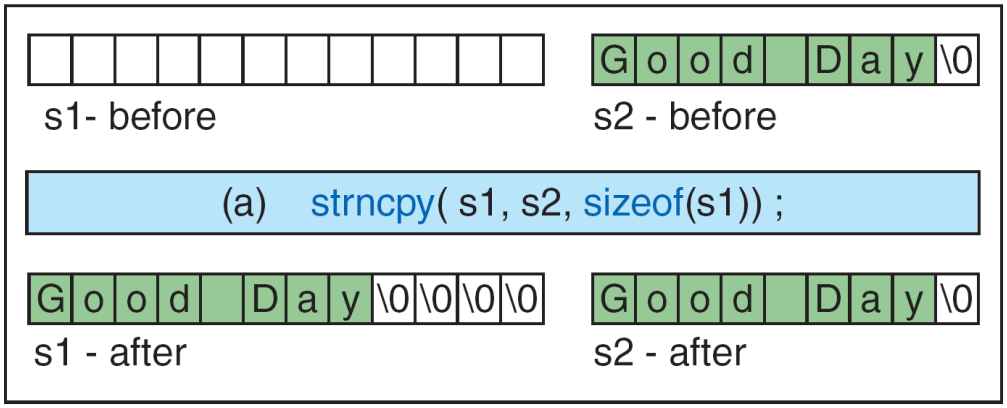
```
char *strncpy (char  *to_string,
               const  char *from_string,
               int    size);
```

Return : to_string

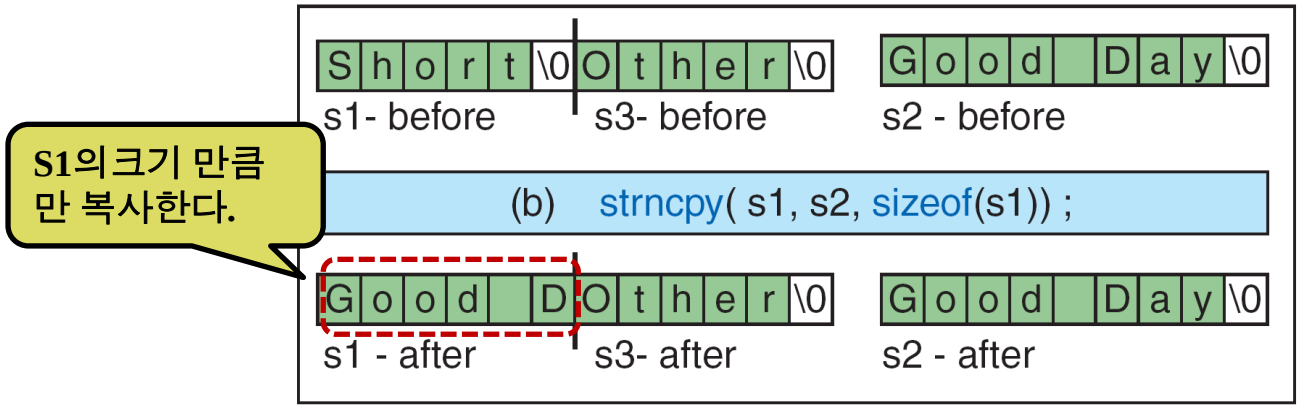
  - from_string에 저장된 문자열을 to_string에 복사한다.
  - size 길이 만큼의 from_string의 문자열을 to_string에 복사한다.
    즉, from_string의 길이에 관계없이 size만큼 복사하기 때문에
    다음과 같은 경우가 생길 수 있다.
    * NULL문자 없이 복사가 종료됨 (if size < strlen(from_string))
    * 뒷부분이 NULL로 채워짐 (if size > strlen(from_string))

String manipulation function

- 다음 그림은 strncpy를 통해서 문자열이 복사되는 모습을 보여준다.



Copying Strings



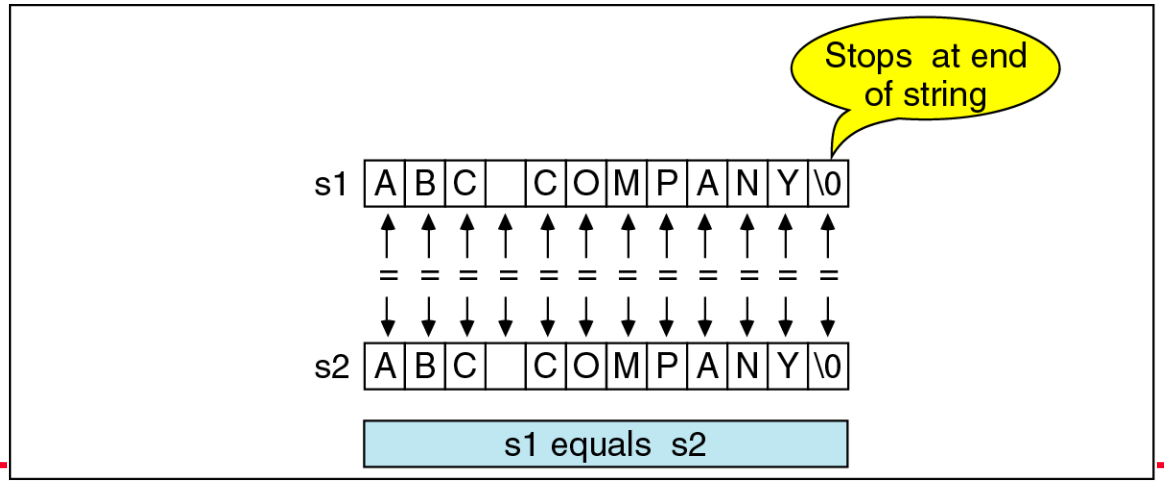
Copying Long Strings

String manipulation function

- **strcmp()**
  - strcmp는 string compare의 약자로서, 이 함수는 두 문자열을 비교한다

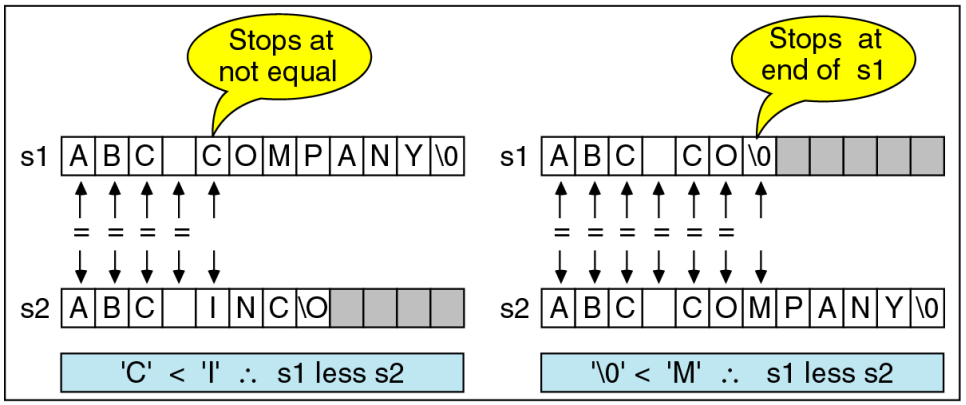
```
int strcmp (const char *string1, const char *string2);
```

  - 두 문자열을 순차적으로 비교하여 string1 이 string2보다 작으면 음수, 크면 양수, 같으면 0을 리턴한다.
  - strcmp(s1, s2)와 같이 호출했을 때, 다음과 같은 경우에는 두 문자열이 서로 같으므로 0을 리턴한다.



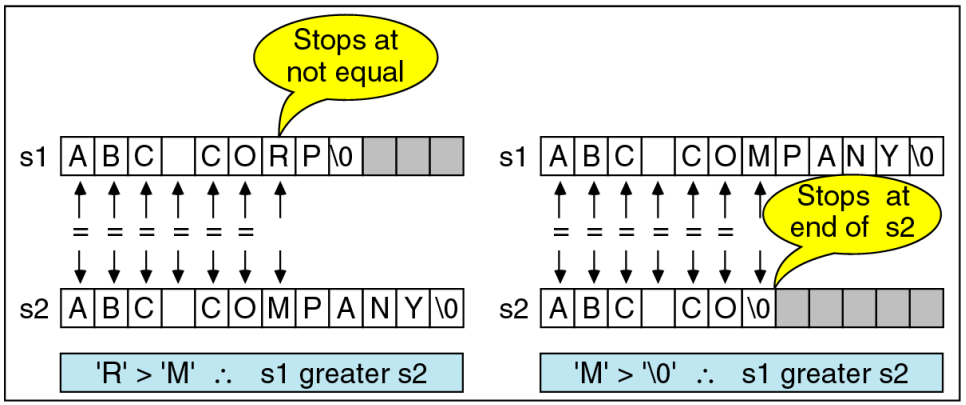
String manipulation function

- 다음은 음수가 리턴되는 경우이다.



- 첫 번째 경우는 'C'가 'I'보다 작다.
- 두 번째 경우는 내용은 서로 같지만 s1이 s2보다 짧다.

- 다음은 양수가 리턴되는 경우이다.



- 첫 번째 경우는 'R'이 'M'보다 크다.
- 두 번째 경우는 내용은 서로 같지만 s2가 s1보다 짧다.

String manipulation function

- **strncmp()**
  - strncmp도 strcmp처럼 두 문자열을 비교하는 함수이다.

```
int strncmp (const char *string1,
             const char *string2,
                   int   size);
```

  - strncmp는 두 문자열을 순차적으로 size의 길이 만큼 비교한다.
  - 비교 결과에 따른 리턴 값은 strcmp와 같다.
    (string1 이 string2보다 작으면 음수, 크면 양수, 같으면 0을 리턴 한다.)

String manipulation function

- strncmp(string1, string2, size); 와 같이 호출했을 때의 리턴 값

string1	string2	Size	Results	Returns
"ABC123"	"ABC123"	8	equal	0
"ABC123"	"ABC456"	3	equal	0
"ABC123"	"ABC456"	4	string1 < string2	< 0
"ABC123"	"ABC"	3	equal	0
"ABC123"	"ABC"	4	string1 > string2	> 0
"ABC"	"ABC123"	3	equal	0
"ABC123"	"123ABC"	–1	equal	0

string1과 string2는 서로 다른 문자열이지만 맨 앞의 3글자만 비교했을 때는 서로 같으므로 0을 리턴한다.

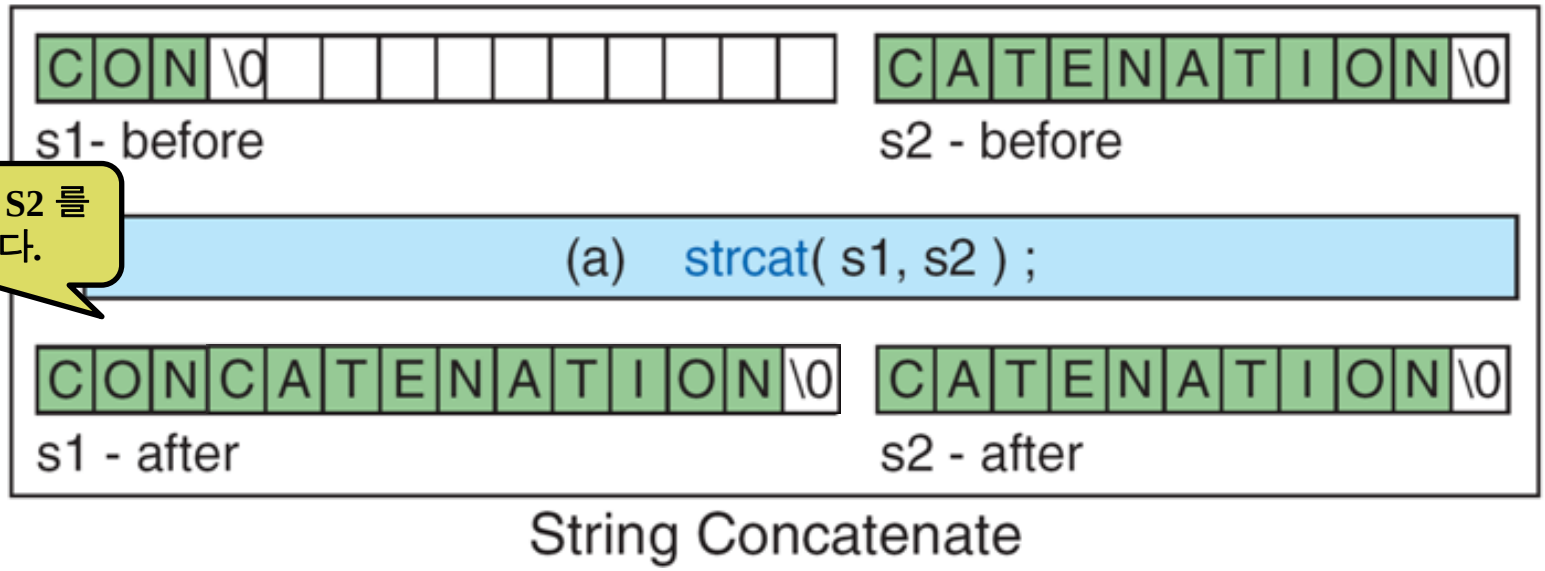
String manipulation function

- **strcat()**
  - **strcat는 string concatenation의 약자로, 이 함수는 두 문자열을 결합한다.**

```
char *strcat (char *string1, const char *string2);
```

Return : string1

  - **string1 의 마지막 '₩0'자리부터 string2의 값을 결합한다.**



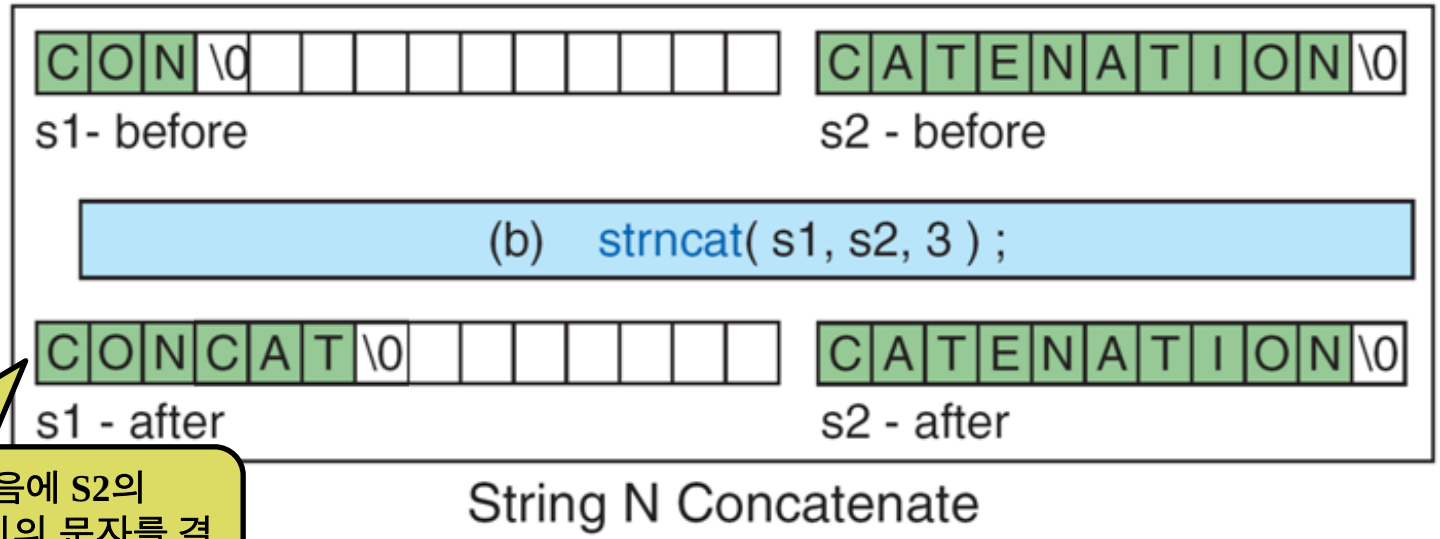
String Concatenate

String manipulation function

- **strncat()**
  - strncat 함수도 strcat 함수처럼 **두 문자열을 결합하는 함수**이다.
  - strncat는 string1 의 마지막 '₩0'자리부터 size 길이 만큼의 string2의 값을 결합한다.

```
char *strncat (        char    *string1,
               const   char    *string2,
                       int      size);
```

Return : string1



S1 다음에 S2의 3번째까지의 문자를 결합한다.

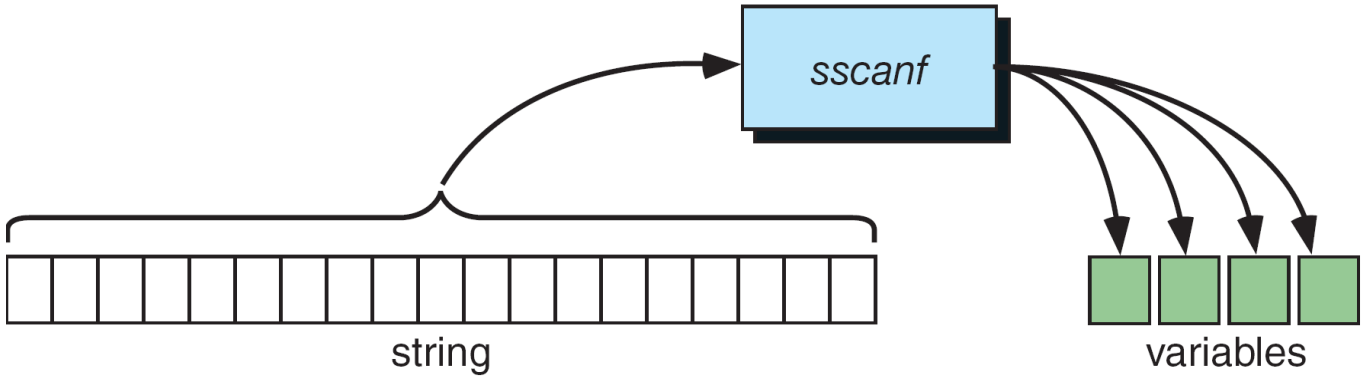
Memory formatting

- **sscanf()**
  - 메모리에 저장된 문자열을 입력으로 받아들여 이를 각 변수에 저장한다.
  - fscanf가 scanf의 파일 버전인 것처럼 sscanf는 scanf()의 메모리 버전이라고 생각할 수 있다.

```
int sscanf (char *str, const char *format_string, …);
```

Return :
성공적으로 읽어들인 데이터 수

  - 실제 사용 방법도 fscanf()와 거의 같다.



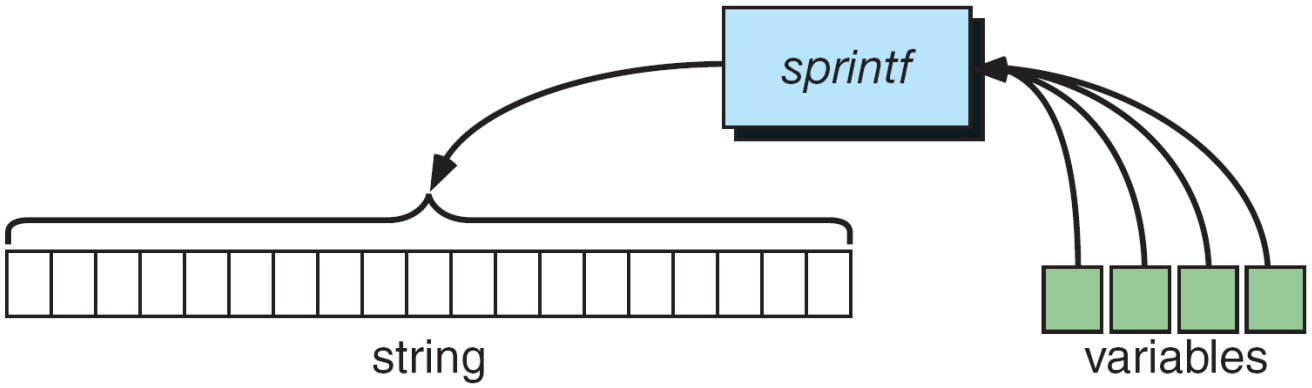
Memory formatting

- **sprintf()**
  - 각 문자열 (또는, 변수들)을 파라미터로 입력 받아 이를 서식에 따라 하나의 문자열로 저장한다 .
  - fprintf가 printf의 파일 버전인 것 처럼, sprintf는 printf의 배열 버전이라고 생각할 수 있다.

```
int sprintf (       char *out_string,
              const char *format_string, …);
```

Return :
NULL을 제외한 출력한 문자 수
Error : EOF

  - 실제 사용 방법도 fprintf와 거의 같다.



Memory formatting

- 예제 프로그램 – sscanf( ) , sprintf( )

```c
#include <stdio.h>

int main(void){

    char str[80] = "Einstein, Albert; 1234 97 A";
    char strOut[80];
    char name[50];
    char id[5];
    int score;
    char grade;
    int n1, n2;

    printf("String contains : \"%s\"\n", str);
    n1 = sscanf(str, "%49[^;] %*c %4s %d %c", name, id, &score, &grade);

    printf("Reformatted data : \n");
    printf("\tName  : \t[%s]\n", name);
    printf("\tID    : \t[%s]\n", id);
    printf("\tScore: \t%d\n", score);
    printf("\tGrade: \t%c\n", grade);

    n2 = sprintf(strOut, "[%s] / %4s / %3d / %c", name, id, score, grade);

    printf("New String :\t\"%s\"\n", strOut);

    printf("n1 = %d, n2 = %d\n", n1, n2);
}
```

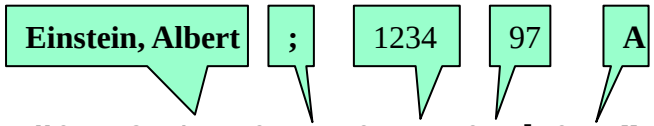
str에 저장되어 있는 내용을 형식에 맞게 입력 받는다.

입력 받은 내용을 원하는 형식으로 바꾸어 출력한다.

```
./a.out
String contains : "Einstein, Albert; 1234 97 A"
Reformatted data :
        Name  :  [Einstein, Albert]
        ID    :  [1234]
        Score:  97
        Grade:  A
New String :    "[Einstein, Albert] / 1234 /  97 / A"
n1 = 4, n2 = 35
```

Memory formatting

- str[80] = "Einstein, Albert; 1234 97 A";



- n1 = sscanf(str, "%49[^;] %*c %4s %d %c", name, id, &score, &grade);

  array str의 내용으로 부터 다음과 같이 형식에 맞게 읽는다

  %49[^;] – ';' 가 입력되기 전까지 최대 49자를 읽어 name에 저장

  %*c – 한 글자를 읽고 그 글자를 무시

  %4s – 4글자를 읽어 id에 저장

  %d – 숫자 하나를 읽고 score에 저장

  %c – 글자 하나를 읽고 grade에 저장

- 성공적으로 읽은 개수를 n1에 저장하므로 n1 = 4가 출력된다.