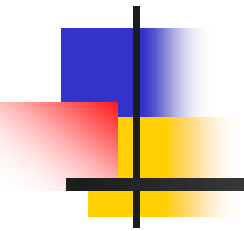


C Programming (CSE2035)
(Chap10. Strings 2)

Sungwon Jung, Ph.D.

Bigdata Processing & DB LAB
Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

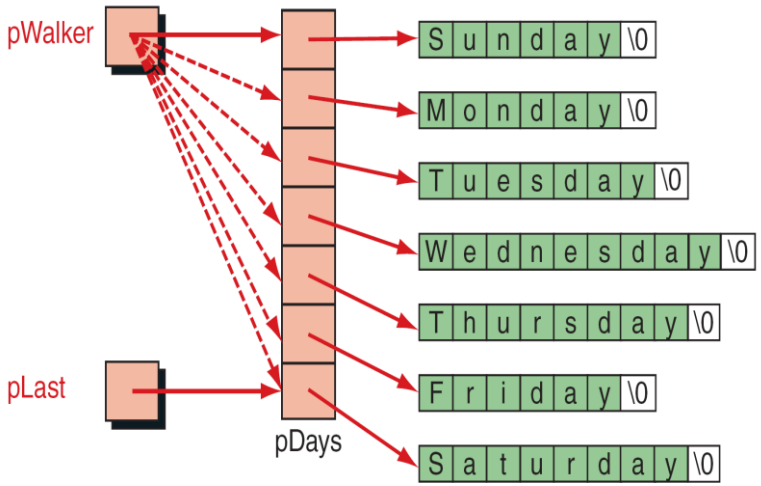
Arrays of strings

- 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

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5          char* pDays[7];
6          char** pLast;
7          char** pWalker;
8
9          pDays[0] = "Sunday";
10         pDays[1] = "Monday";
11         pDays[2] = "Tuesday";
12         pDays[3] = "Wednesday";
13         pDays[4] = "Thursday";
14         pDays[5] = "Friday";
15         pDays[6] = "Saturday";
16
17         printf("The days of the week\n");
18         pLast = pDays + 6;
19
20         for(pWalker=pDays; pWalker <= pLast; pWalker++)
21                 printf("%s\n", *pWalker);
22
23         return 0;
24 }
25
```

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

pDays의 내용을 순차적으로 출력



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 이전까지의 문자 개수를 의미한다.

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

Return : *string부터 처음 만나는 '\0' 이전까지의 길이

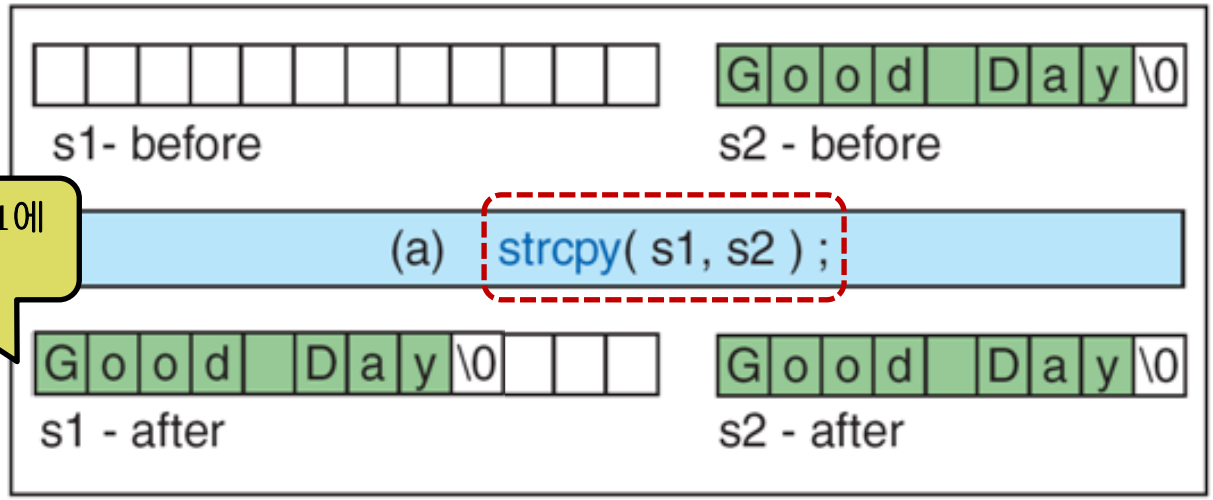
String manipulation function

- **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.



S2의 문자열이 S1에 복사된다

Copying Strings

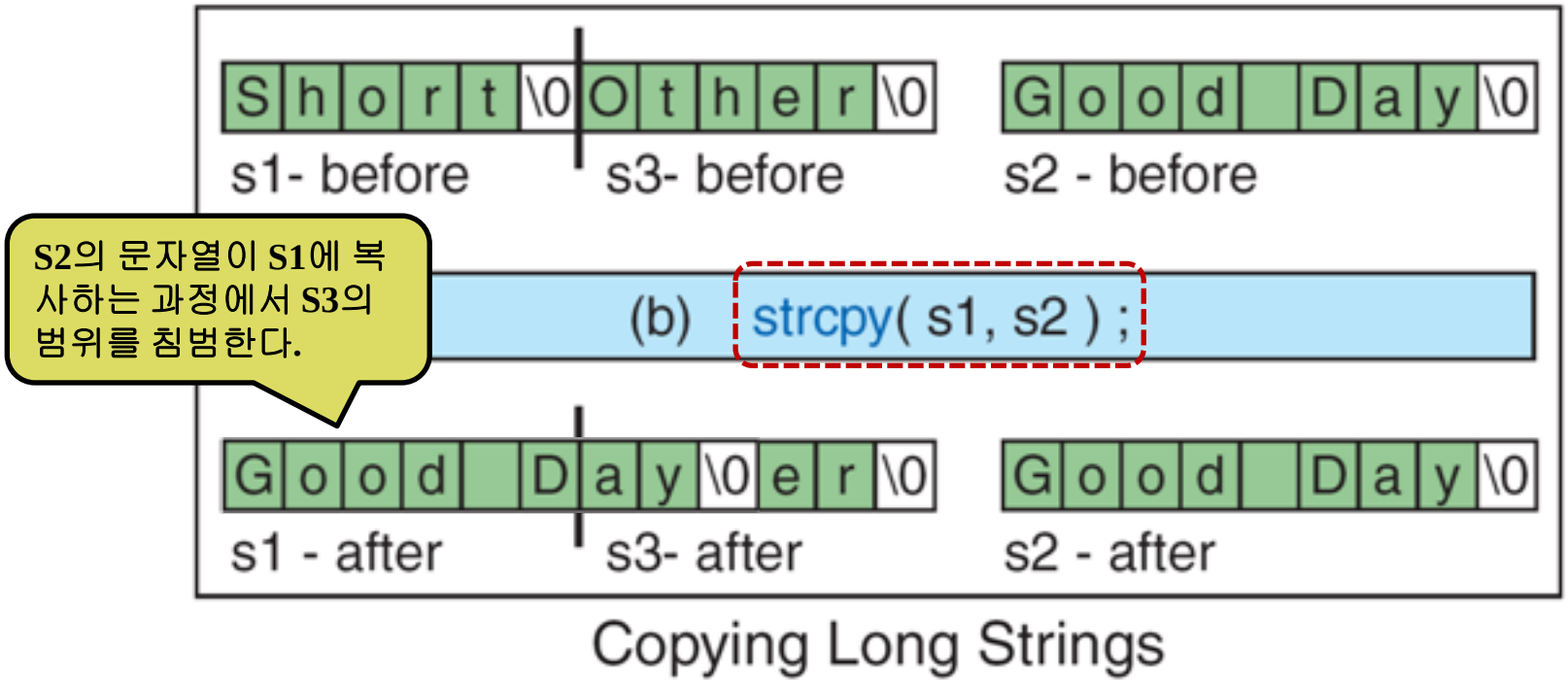
String manipulation function

- **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.

- 참고
  char* stpcpy(char *dest, const char *src); 는 strcpy와 기능상 동일하나, dest + strlen(src)의 값을 return한다.

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

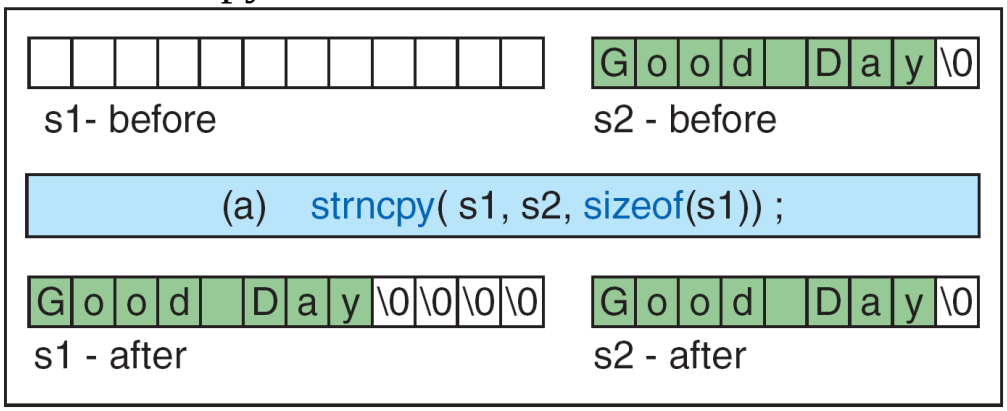
```
char *strncpy (char  *to_string,
               const  char *from_string,
               int    size);
```

Return : to_string

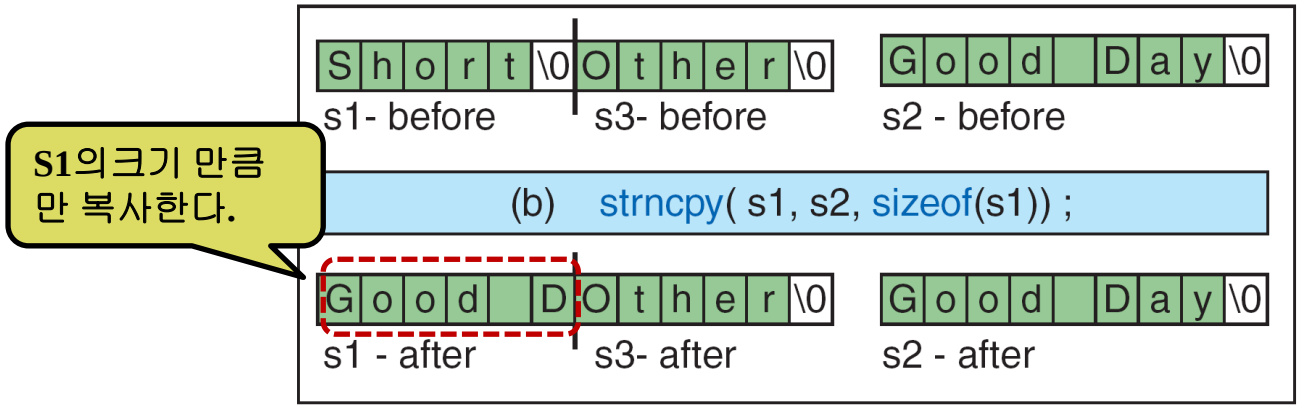
  - from_string에 저장된 문자열을 to_string에 복사한다.
  - size 길이 만큼의 from_string의 문자열을 to_string에 복사한다.
    즉, from_string의 길이에 관계없이 size만큼 복사하기 때문에
    다음과 같은 경우가 생길 수 있다.
    * NULL문자 없이 복사가 종료됨 (if size < strlen(from_string))
    * 뒷부분이 NULL로 채워짐 (if size > strlen(from_string))

String manipulation function

- 다음 그림은 strncpy를 통해서 문자열이 복사되는 모습을 보여준다.



Copying Strings



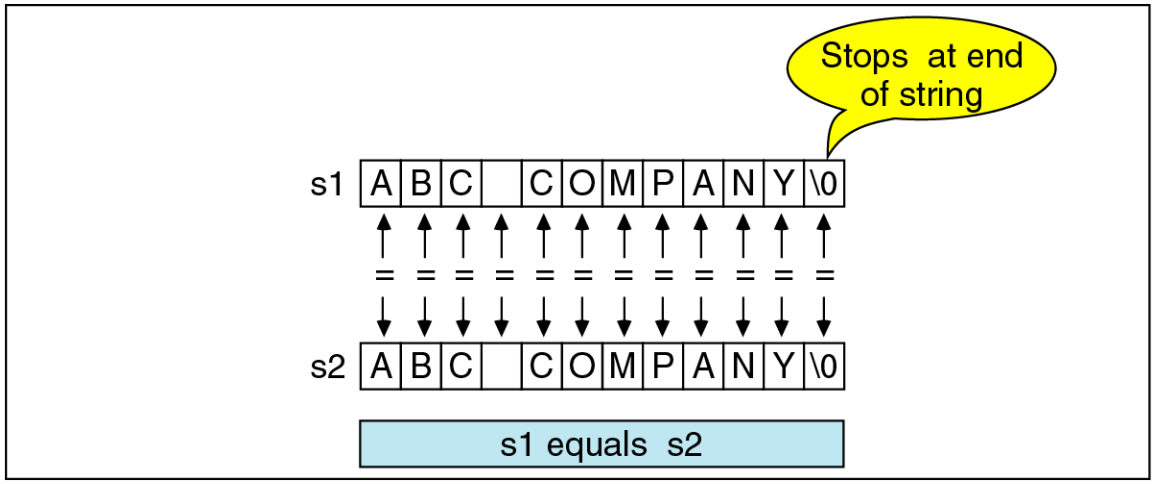
Copying Long Strings

String manipulation function

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

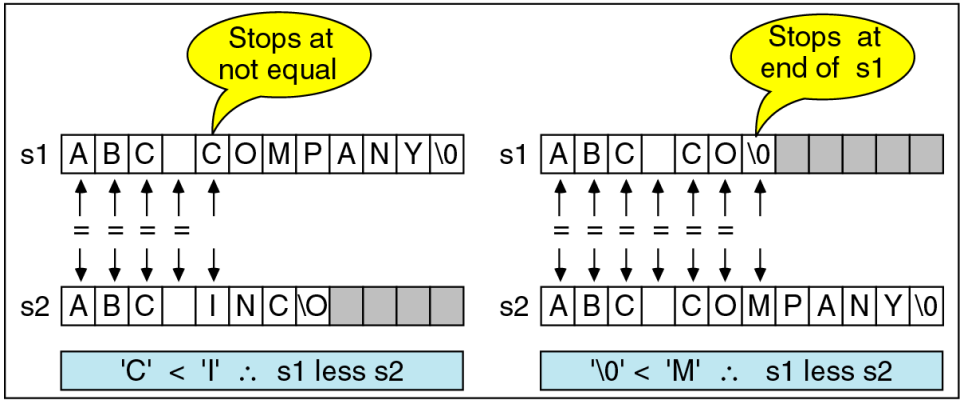
```
int strcmp (const char *string1, const char *string2);
```

  - 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.



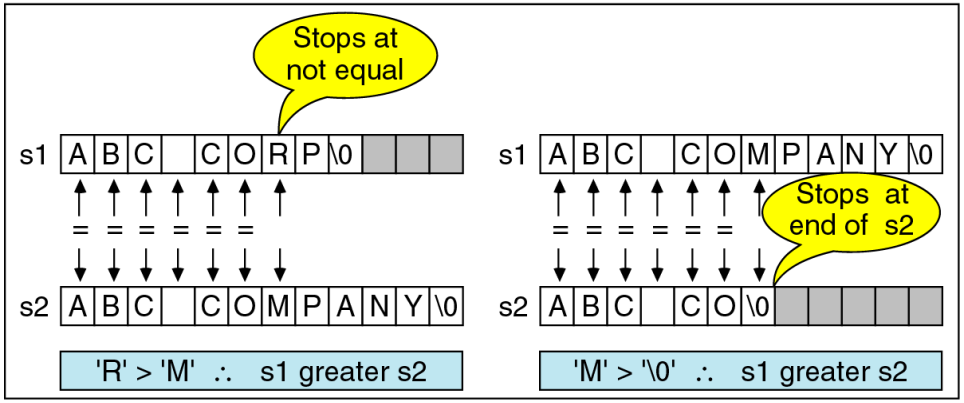
String manipulation function

- 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 작다.
- 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.

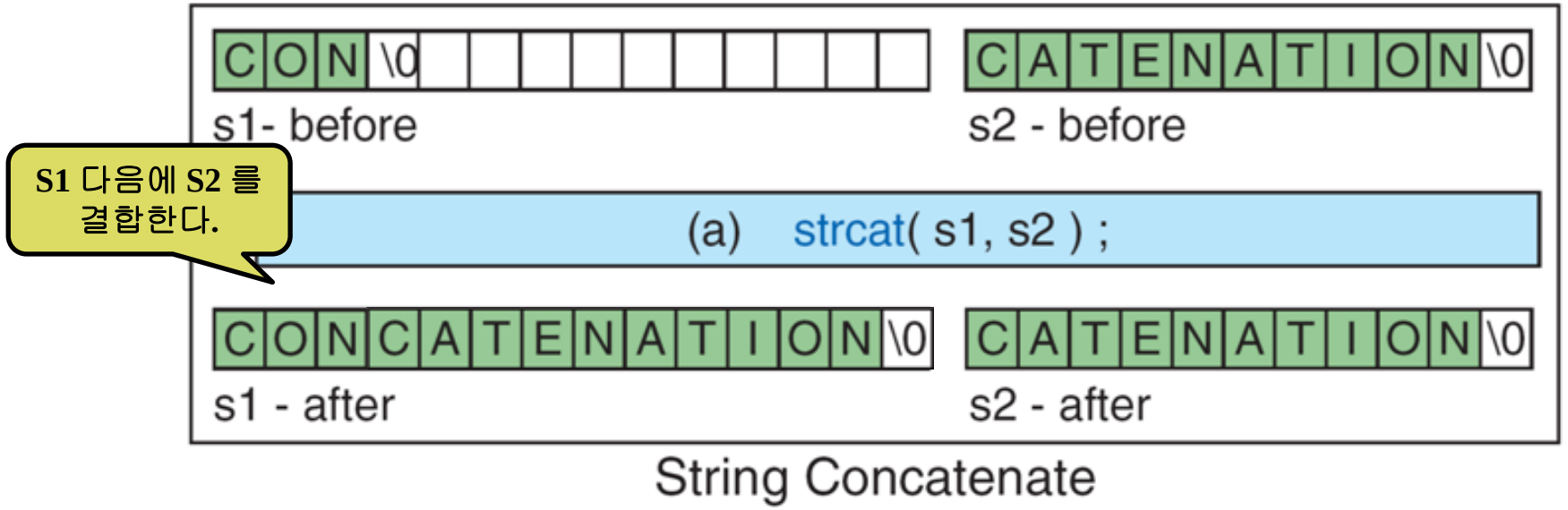
String manipulation function

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

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

Return : string1

  - string1 의 마지막 '\0'자리부터 string2의 값을 결합한다.



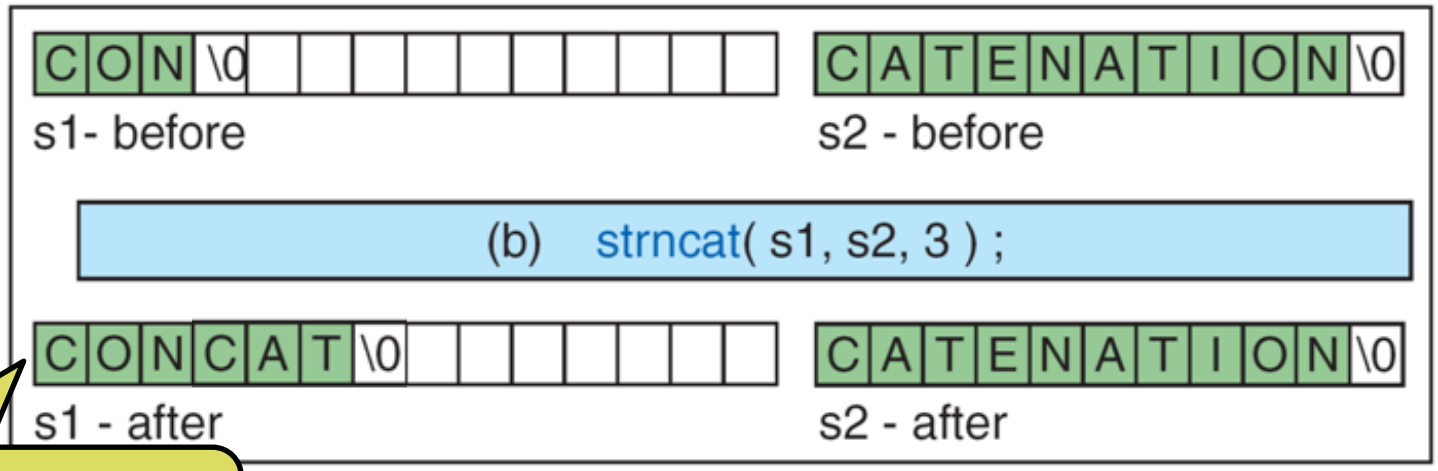
String Concatenate

String manipulation function

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

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

Return : string1



String N Concatenate

S1 다음에 S2의 3번째까지의 문자를 결합한다.

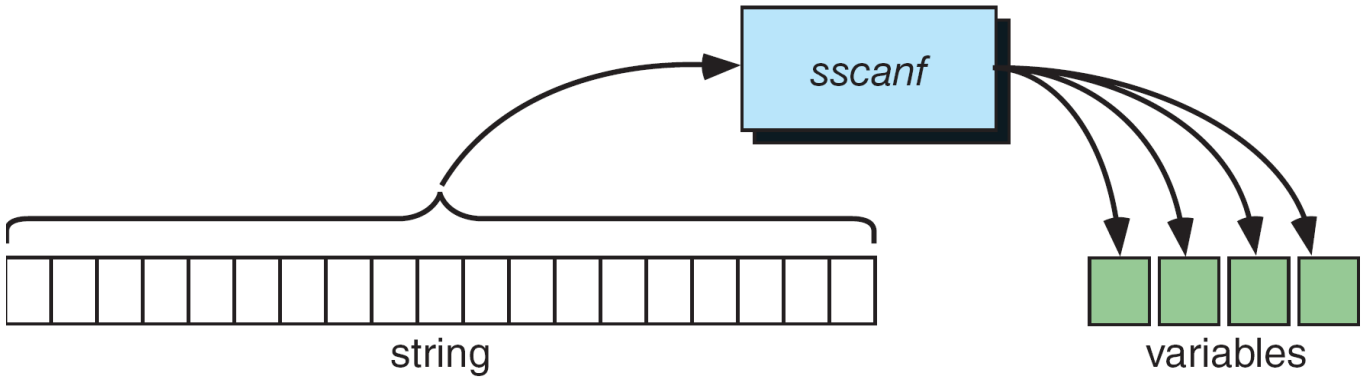
Memory formatting

- **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.



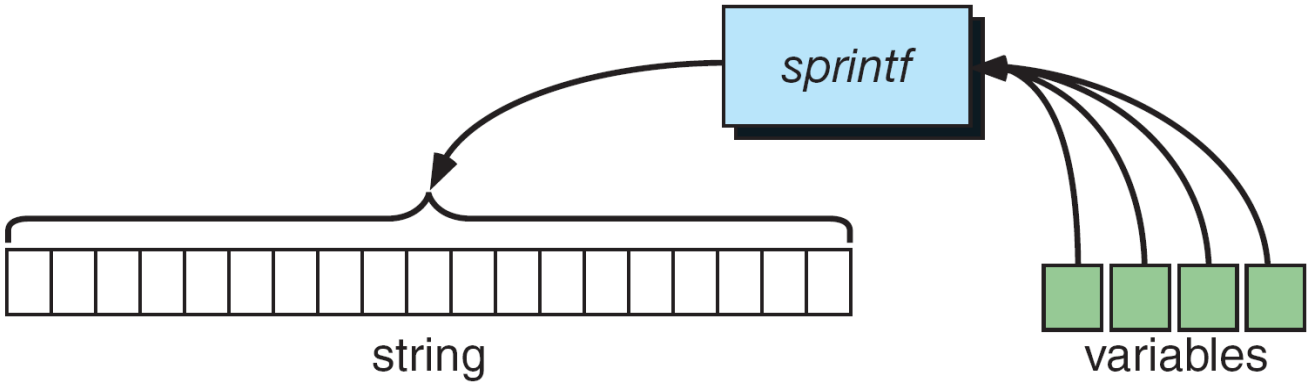
Memory formatting

- **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);
}
```

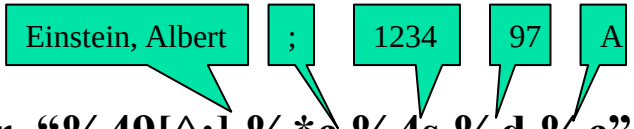
str에 저장되어 있는 내용을 형식에 맞게 입력 받는다.

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 **49**자를 읽어 **name**에 저장
  - **%*c** – 한 글자를 읽고 그 글자를 무시
  - **%4s** – **4**글자를 읽어 **id**에 저장
  - **%d** – 숫자 하나를 읽고 **score**에 저장
  - **%c** – 글자 하나를 읽고 **grade**에 저장

- 성공적으로 읽은 개수를 **n1**에 저장하므로 **n1 = 4**가 출력된다.