C 기초 특강

문자열

문자열 (1)

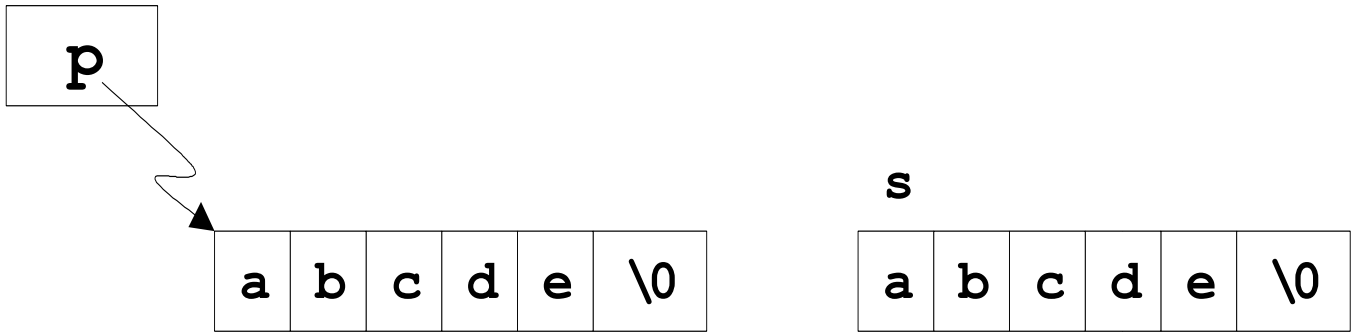
- 문자열
  - char 형의 1차원 배열
  - 문자열은 끝의 기호인 \0(널 문자)로 끝남
  - 널 문자 : 모든 비트가 0인 바이트; 십진 값 0
  - 문자열의 크기는 \0까지 포함한 크기
- 문자열 상수
  - 큰따옴표 안에 기술됨
  - 문자열 예 : "abc"
    - 마지막 원소가 널 문자이고 크기가 4인 문자 배열
- 주의 - "a"와 'a'는 다름
  - 배열 "a"는 두 원소를 가짐
  - 첫 번째 원소는 'a', 두 번째 원소는 '\0'

문자열 (2)

- 컴파일러는 문자열 상수를 배열 이름과 같이 포인터로 취급
  - char *p = "abc";
  - printf("%s %s\n", p, p + 1); /* abc bc is printed */
  - 변수 p에는 문자 배열 "abc"의 기본 주소가 배정
  - char 형의 포인터를 문자열 형식으로 출력하면, 그 포인터가 포인트하는 문자부터 시작하여 \0이 나올 때까지 문자들이 연속해서 출력됨
- "abc"와 같은 문자열 상수는 포인터로 취급되기 때문에 "abc"[1] 또는 *("abc" + 2)와 같은 수식을 사용할 수 있음

문자열 (3)

- 배열과 포인터의 차이
  - char *p = "abcde";
  - char s[ ] = "abcde";
    - /* char s[ ] = 'a', 'b', 'c', 'd', 'e', '\0'}; */



문자열 (4)

- 예제 코드

```
#include   <ctype.h>
int word_cnt(const char *s)
{
   int    cnt = 0;
   while (*s != '\0') {
      while (isspace(*s))                    /* skip white space */
         ++s;
      if (*s != '\0') {                      /* found a word */
         ++cnt;
         while (!isspace(*s) && *s != '\0') /* skip the word */
             ++s;
      }
   }
   return cnt;
}
```

문자열 조작 함수 (1)

- char *strcat(char *s1, const char *s2);
  - 두 문자열 s1,s2를 결합하고, 결과는 s1에 저장
- int strcmp(const char *s1, const char *s2);
  - s1과 s2를 사전적 순서로 비교하여, s1이 작으면 음수, 크면 양수, 같으면 0을 리턴
- char *strcpy(char *s1, const char *s2);
  - s2의 문자를 \0이 나올 때까지 s1에 복사
- size_t strlen(const char *s);
  - \0을 뺀 문자의 개수를 리턴

문자열 조작 함수 (2)

- 구현 방법

```
size_t strlen(const char *s)
{
    size_t   n;
    for (n = 0; *s != '\0'; ++s)
       ++n;
    return n;
}
```

```
char *strcpy(char *s1, const char *s2)
{
    register char   *p = s1;
    while (*p++ = *s2++)
       ;
    return s1;
}
```

문자열 조작 함수 (3)

선언 및 초기화	
`char s1[ ] = "beautiful big sky country",` `s2[ ] = "how now brown cow";`	
수식	값

`strlen(s1)`
`strlen(s2 + 8)`
`strcmp(s1, s2)`
	25
	9
	음의 정수

문장	출력되는 값

`printf("%s", s1+10);`
`strcpy(s1 + 10, s2 + 8);`
`strcat(s1, "s!");`
`printf("%s", s1);`
	big sky country
	
	beautiful brown cows!

안전한 문자열 관련 함수 사용 (1)

- C 언어에서는 프로그래머에게 배열 관리 책임이 있음
  - 실험 코드 1

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

int main()
{
        char bad_str[15];

        strcpy(bad_str, "short string");
        printf("%s\n", bad_str);

        strcpy(bad_str, "very very very long string");
        printf("%s\n", bad_str);
}
```

```
[kgu@lily str]$ ./a.out
short string
very very very long string
세그멘테이션 오류 (core dumped)
```

안전한 문자열 관련 함수 사용 (2)

- 실험 코드 2

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

int main()
{
        char bad_str1[15];
        char bad_str2[15];

        strcpy(bad_str1, "12345678901234");
        strcpy(bad_str2, "short string");
        printf("%s\n", bad_str1);
        printf("%s\n", bad_str2);

        strcpy(bad_str2, "very very very long string");
        printf("%s\n", bad_str1);
        printf("%s\n", bad_str2);
}
```

```
[kgu@lily str]$ ./a.out
12345678901234
short string
ong string
very very very long string
```

안전한 문자열 관련 함수 사용 (3)

- 실험코드 3

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

#define GOOD_STR_LENGTH 15

int main()
{
        char good_str[GOOD_STR_LENGTH + 1];

        strncpy(good_str, "short string",
                        GOOD_STR_LENGTH);
        printf("%s\n", good_str);

        strncpy(good_str,
                         "very very very long string",
                        GOOD_STR_LENGTH);
        printf("%s\n", good_str);
}
```

안전한 문자열 관련 함수 사용 (4)

- 버퍼 오버플로우(buffer overflow) 공격
- 안전한 문자열 관련 함수
  - strcpy -> strncpy
  - strcat -> strncat