

Strings en C

Edgardo Hames

Introducción

- Una variable de tipo *char* sólo puede almacenar un único carácter.
- Un *string* es una secuencia de caracteres. Por ej: "Hallo, Welt!"
- Una mala noticia: C no soporta el tipo *string*.

Arrays de caracteres

- En C un *string* se maneja como un array de caracteres.
- Generalmente, una variable con valor de tipo *string* se declara como un puntero de tipo *char **.

```
char *saludo = "Salut, mundi.";
```

- Por convención, un carácter nulo `\0` marca el fin de un string.

```
char *mensaje = {'H','o','l','a','\0'};
```

```
char *ptr;
```

```
if (!*ptr) {  
    /* p apunta a un carácter nulo */  
}
```

```
if (*ptr == '\0') {  
    /* p apunta a un carácter nulo */  
}
```

- Un carácter nulo es conceptualmente distinto de NULL.

- Arrays de caracteres declarados como *const* no pueden ser modificados. Por lo tanto, es de muy buen estilo declarar strings no modificables de tipo *const char **.

La destreza y la memoria son buenas si van en yunta

- La memoria asignada para un array de caracteres puede extenderse más allá del caracter nulo.

```
char ptr[20];

strcpy(ptr, "Hola, Mundo");
/*Sobran 2^3 caracteres: ptr[12] - ptr[19] */
```

- Una fuente muy común de bugs es intentar poner más caracteres de los que caben en el espacio asignado. Recordemos hacer lugar para `\0` !
- Las funciones de la biblioteca NO toman en cuenta el tamaño de la memoria asignada. C asume que el programador sabe lo que hace...

```
char a[10], b[10];

strcpy(a, "Por el Río Paraná...");
printf("b = %s\n", b); /* Auch!!! */
```

- Para muchas funciones de la forma `strXXX` existe una versión `strnXXX` que opera sobre los *n* primeros caracteres del array.

Usando `p`, `*p`, `p[]` (1)

- Apuntando a un string.

```
char *ptr = "Hello, World!";
```

- Es lo mismo que hacer

```
char ptr[] = "Hello, World!";
```

- Para acceder al string “Hello, World!” usamos la variable `ptr`.

```
printf("String: %s\n", ptr);
```

- Cada uno de los caracteres es accedido indexando la variable `ptr`.

```
printf("Char: %c\n", ptr[1]);
```

- Notar que podemos usar un índice mayor a la longitud del string sin que nos dé error.

```
/* Probar en Haskell: "Hello, World!" !! 20 */
printf("Char: %c\n", ptr[20]);
```

Usando `p`, `*p`, `p[]` (2)

- PUNTERO A UN CHARACTER

```
char *p, a;

a = 'A';
p = &a;
printf("Contenido de p = %c\n", *p); /* Imprime A */
a = 'B';
printf("Contenido de p = %c\n", *p); /* Imprime B */
```

- ARRAY Y PUNTERO A UN ARRAY

```
char p[5] = {'H', 'o', 'l', 'a', '\0'};

printf("Un caracter = %c\n", p[0]); /* Imprime H */
printf("Contenido de p = %s\n", p); /* p apunta al array */
```

Funciones útiles (1)

- Estas funciones operan sobre caracteres:

```
#include <ctype.h>

int isalnum (int c);
int isalpha (int c);
int isascii (int c);
int isblank (int c);
int iscntrl (int c);
int isdigit (int c);
int isgraph (int c);
int islower (int c);
int isprint (int c);
int ispunct (int c);
int isspace (int c);
int isupper (int c);
int isxdigit (int c);
```

Ver sección 3 del manual.

Funciones útiles (2)

- Estas funciones operan sobre strings:

```
#include <string.h>

char *strcpy(char *dest, const char *orig);
char *strcat(char *dest, const char *src);
char *strstr(const char *haystack, const char *needle);
size_t strspn(const char *s, const char *accepta);
size_t strcspn(const char *s, const char *rechaza);
```

Ver sección 3 del manual.

Referencias

- <http://www.harpercollege.edu/bus-ss/cis/166/mmckenzi/contents.htm>
- <http://www.harpercollege.edu/bus-ss/cis/166/mmckenzi/lect12/l12.htm>
- <http://bstring.sourceforge.net>
- <http://www.cs.princeton.edu/courses/archive/spring02/cs217/asgts/ish/ish.html>
- <http://www.cs.princeton.edu/courses/archive/spring02/cs217/asgts/ish/ishhints.html>