L16 - Strings

November 22, 2019

1 Strings

Unlike Python and C++, C does *not* have a string type. Strings are implemented as arrays of characters, terminated with a `\0` or *null character*. Upon declaration, the C compiler will create *and initialize* the array.

Note that the null character is not the same as the "0" character or the `NULL` null pointer.

The number of elements in a character array is one more than the number of characters it is initialized with, to accomodate the null character.

C does not allow assinging a string literal into a character array.

As with other variables, the `const` keyword tells the compiler that the variable is *immutable*. Attempting to write into or modify a `const`'d variable will generate an error.

C operators are not overloaded to allow for string concatenation - you cannot `+` two strings together.

1.1 `strlen()`

`size_t strlen(const char *s)`

Returns the length of a string pointed to by s, not including the terminating null.

```
In [36]: #include <string.h>
         #include <stdio.h>

         void main()
         {
             char greeting[] = "Hello";
             size_t len;
             len = strlen(greeting);
             // returns 5 (not 6)
             printf("%zd", len); // use 'z' modifier to turn size_t to int, I guess
         }
```

```
5
```

1.2 `strcmp()`

`int strcmp(const char *s, const char *t)`

Compares the string pointed to by s to the string pointed to by t. Specifically, it returns whether the sum of the ASCII encodings in s is greater than t.

Returns a positive value if `s > t`, zero if `s == t`, and a negative value if `s < t`.

```
In [ ]: char name1[30];
        char name2[30];

        void main()
        {
            // Initialization of name1 and name2
            // not shown
            if (strcmp(name1, name2) != 0) {
            // strings are different
            }
        }
```

1.3 `sprintf()`

Similar to `printf()`, but stores the output in the first argument. Note that this will *not* output to the console.

```
In [17]: #include <string.h> // library for fun string stuff
         #include <stdio.h>
         #include <stdlib.h>

         void main()
         {
             const char str1[] = "SYSC2006";
             const char str2[] = "F19";

             printf("%s %s\n", str1, str2);

             char msg[50];

             sprintf(msg, "%s %s", str1, str2);

             printf("%s\n", msg);
         }
```

```
SYSC2006 F19
SYSC2006 F19
```

1.4 `sscanf()`

Reads from the first argument, which should be a string. Can be used to parse function arguments.

```
In [23]: #include <string.h>
         #include <stdio.h>
         #include <stdlib.h>
```

```
        void main()
        {
            char msg1[] = "SYSC2006 2019";

            printf("msg1 = %s\n", msg1);

            int year;
            char str1[22];

            sscanf(msg1, "%s %d", str1, &year);

            printf("str1 = %s, year = %d\n", str1, year);
        }
```

```
msg1 = SYSC2006 2019
str1 = SYSC2006, year = 2019
```

1.5 TCP/IP Network Protocol Emulation

A 4-tuple gets attached to a packet on the network: * source IP address

- destination IP address
- source port number
- destination port number

In [26]:
```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

// Encode a message ( mostly from the slides )

char src[] = "10.1.1.1"; // Src IP addr
char dst[] = "10.2.2.2"; // Dest IP addr
int tos = 4; // Type of Service
int msgLen = 200; // packet length

void main()
{
    // somewhere to put our message
    char msg[100];

    // semicolon separated string thing
    sprintf(msg, "%s;%s;%d;%d", src, dst, tos, msgLen);

    printf("msg = %s\n", msg);
}
```

```
msg = 10.1.1.1;10.2.2.2;4;200
```

In [32]:
```c
// Decode (parse) a message ( also mostly from the slides )
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

char src[20];
char dst[20];
int tos;
int len;

// parse the message into several variables
void main()
{
    // a message to decode
    char msg[] = "10.1.1.1;10.2.2.2;4;200";

    sscanf(msg, "%[^;];%[^;];%d;%d", src, dst, &tos, &len);

    printf("src = %s, dst = %s, tos = %d, len = %d\n\n", src, dst, tos, len);
}
```

```
src = 10.1.1.1, dst = 10.2.2.2, tos = 4, len = 200
```