[(79 条消息) C 语言—字符串与字符数组，字符串数组与字符串指针的区别_西海岸看日出的博客-CSDN 博客_c 字符数组与字符串]

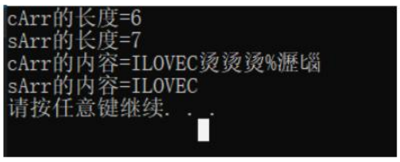
字符串与字符数组

字符串一定是一个char的数组，但char的数组未必是字符串。——以字符'\0'结尾的char数组就是一个字符串，但如果char数组没有以'\0'结尾，那么就不是一个字符串，只是普通字符数组，所以**字符串是一种特殊的char数组**。

两者的区别：
字符串：
1、字符串赋初值用双引号引起来；
2、以隐含的空字符\0结束，占用字节数+1，注意：1字节/字母，2字节/汉字；
3、字符串可以使用%s格式化输出。
普通字符数组：
1、普通的字符数组赋初值用大括号引起来；
2、不包含空字符，占用字节数不需+1；
3、普通字符数组使用%s格式化输出，输出结果会**乱码**，因为没有'\0'结束符。
例如：

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    /*字符数组赋初值*/
    char cArr[] = {'I','L','O','V','E','C'};
    /*字符串赋初值*/
    char sArr[] = "ILOVEC";
    /*用sizeof（）求长度*/
    printf("cArr的长度=%d\n", sizeof(cArr));
    printf("sArr的长度=%d\n", sizeof(sArr));
    /*用printf的%s打印内容*/
    printf("cArr的内容=%s\n", cArr);
    printf("sArr的内容=%s\n", sArr);
    return 0;
}
```

运行结果为：



如果定义：

```
char cArr[] = {'I','L','O','V','E','C','\0'};
char sArr[] = "ILOVEC";
```

那么cArr与sArr就完全相同，都表示同一字符串。

另外，%s格式化输出从当前地址到结束标志'\0'之前的所有字符，如：

```c
#include <stdio.h>
#include <string.h>
int main(){
    char str[] = "http://c.biancheng.net";
    printf("%s\n", str); //直接输出字符串
    printf("%s\n", str+1); //输出从第二个地址到最后的值
    printf("%s\n", str+2); //输出从第三个地址到最后的值
    return 0;
}
```

```
http://c.biancheng.net
ttp://c.biancheng.net
tp://c.biancheng.net
请按任意键继续. . .
```

通过对以上代码的分析，现在我们可以很简单地总结出字符数组和字符串二者之间的区别：
对于字符数组，其长度是固定的，其中任何一个数组元素都可以为 null 字符。因此，字符数组不一定是字符串。
对于字符串，它必须以 null 结尾，其后的字符不属于该字符串。字符串一定是字符数组，它是最后一个字符为 null 字符的字符数组。

字符串数组与字符串指针

字符串数组：

```c
//字符串数组
#include <stdio.h>
#include <string.h>
int main(){
    char str[] = "http://c.biancheng.net";
    int len = strlen(str), i;
    //直接输出字符串
    printf("%s\n", str); //可以直接输出当前地址（字符串首地址）到结束标志'\0'之前的所有字符
    //每次输出一个字符
    for(i=0; i<len; i++){
        printf("%c", str[i]);
    }
    printf("\n");
    return 0;
}
```

字符串指针：

```c
//字符串指针
#include <stdio.h>
#include <string.h>

int main(){
char *str = "http://c.biancheng.net";
int len = strlen(str), i;

//直接输出字符串
printf("%s\n", str); //可以直接输出当前地址（字符串首地址）到结束标志'\0'之前的所有字符
//使用*(str+i)
for(i=0; i<len; i++){
printf("%c", *(str+i));
}
printf("\n");
//使用str[i]
for(i=0; i<len; i++){
printf("%c", str[i]);
}
printf("\n");

return 0;
}
```

运行结果都是：
http://c.biancheng.net

这一切看起来和字符数组是多么地相似，它们都可以使用%s输出整个字符串，都可以使用*或[ ]获取单个字符，这两种表示字符串的方式是不是就没有区别了呢？

有！它们最根本的区别是在内存中的存储区域不一样，字符数组存储在全局数据区或栈区，第二种形式的字符串存储在常量区。全局数据区和栈区的字符串（也包括其他数据）有读取和写入的权限，而常量区的字符串（也包括其他数据）只有读取权限，没有写入权限。

内存权限的不同导致的一个明显结果就是，字符数组在定义后可以读取和修改每个字符，而对于第二种形式的字符串，一旦被定义后就只能读取不能修改，任何对它的赋值都是错误的。

我们将第二种形式的字符串称为字符串常量，意思很明显，常量只能读取不能写入。请看下面的演示：

```
1
2  #include <stdio.h>
3  int main(){
4  char *str = "Hello World!";
5  str = "I love C!"; //正确
6  str[3] = 'P'; //错误
7
8  return 0;
9  }
```

这段代码能够正常编译和链接，但在运行时会出现段错误（Segment Fault）或者写入位置错误。
第4行代码是正确的，可以更改指针变量本身的指向；第5行代码是错误的，不能修改字符串中的字符。

到底使用字符数组还是字符串常量
在编程过程中如果只涉及到对字符串的读取，那么字符数组和字符串常量都能够满足要求；如果有写入（修改）操作，那么只能使用字符数组，不能使用字符串常量。

最后我们来总结一下，C语言有两种表示字符串的方法，一种是字符数组，另一种是字符串常量（用指针表示），它们在内存中的存储位置不同，使得字符数组可以读取和修改，而字符串常量只能读取不能修改。