

# 第八章 字符串

- 一、字符串的定义
- 二、字符串初始化
- 三、字符串操作函数
- 四、字符串数组

## 一、字符串的定义

C语言中的字符串是由字符数组构成的并且以'\0'作为结束符。字符串用""双引号包围起来，如"Hello world"。

字符数组：数组中的每一个元素都是字符，`char c[4] = {'T', 'h', 'i', 's'};`但它并不是一个字符串，因为它的最后一个字符并不是'\0'。

对于"Hello world"这样的字符串，它在内存中是这样表示的：

H	e	l	l	o	w	o	r	l	d	\0
---	---	---	---	---	---	---	---	---	---	----

字符数组与字符串的区别：字符数组不需要以\0结束，但字符串必须以\0结束。

## 二、字符串的初始化

- 最复杂的方式:

```
char str[11] = {'G','o','o','d',' ','D','a','y','!','\0'};
```

以字符数组的形式,一个字符一个字符的初始化。

- 相对简单的方式:

```
char str[11] = "Good Day!";
```

用一个字符串来初始化一个字符数组。

或者: `char str[ ] = "Good Day!";` 省略下标。

- 另外一种初始化方式:

```
char* pstr = "Good Day!";
```

把一个字符串赋给一个字符指针,则这个指针指向字符串的首元素,且这样的字符串是一个字符串常量,即"Good Day!"存储在常量区,其内容不能被修改,等价于`const char* pstr = "Good Day!";`

结论:

根据字符串存储在内存中的位置划分，字符串主要有三种存储方式，

方式一：栈区，字符串的内容可以被改变

```
char str[11] = "Good Day!";
```

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    char str[11] = "Good Day!";
    str[0] = 'g';           //将字符串的首字符'G'改为'g'
    printf("%s\n",str);
    return 0;
}
```

程序运行结果如下：  
good Day!

方式二：常量区，字符串的内容不可以被改变

```
char* pstr = "Good Day!"; //缺省const
```

```
#include <stdio.h>

int main(int argc, const char * argv[]) {
    char* str = "Good Day!";
    str[0] = 'g';           //试图将字符串的首字符'G'改为'g'
    printf("%s\n", str);
    return 0;
}
```

Thread 1: EXC\_BAD\_ACCESS (code=2, address=0x100000f8a)

程序执行到 `str[0] = 'g';` 时崩溃。

方式三：堆区，使用malloc函数在堆区开辟一块空间，再将字符串存入，字符串的内容可以被改变

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

int main(int argc, const char * argv[]) {
    char *str = (char*)malloc(20);
    //将字符串的内容一个一个的拷贝到堆空间中
    strcpy(str, "Helloworld");
    printf("%s\n", str);

    str[0] = 'h';
    printf("%s\n", str);

    free(str);
    return 0;
}
```

程序运行结果如下：  
Helloworld

程序运行结果如下：  
helloworld

### 三、字符串操作函数

C语言中有很多用于操作字符串的函数，它们定义在<string.h>头文件中，如：

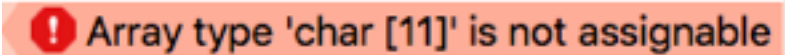
strcpy(str1, str2) 复制拷贝字符串

strcmp(str1, str2) 比较两个字符串

strlen(str) 求字符串长度

strcat(str1, str2) 追加字符串

Example: 试图将str1的内容赋值给str2

```
char str1[11] = "Hello";  
char str2[11];  
str2 = str1; 
```

数组名是常量不能被赋值，如果想赋值必须一个一个的将str1的内容拷贝给str2

```
for (int i = 0; i<11; i++) {  
    str2[i] = str1[i]; //将str1的每一个字符一个一个的拷贝到str2中  
}
```

上述拷贝存在两个问题：

1. 拷贝过程比较麻烦;
2. 字符串1的长度实际只有6个字符（包括\0），但for循环的次数使用了11;

- Function : `strlen` (length) 返回字符串的字符数（长度），不包括\0

函数原型：

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

如何使用？

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

int main(int argc, const char * argv[]) {
    char *pstr="Hello";
    int len = strlen(pstr);
    printf("字符串的长度为:%d\n",len);
    return 0;
}
```

程序运行结果如下：  
字符串的长度为：5



- Function : strcpy 字符串拷贝函数, from源字符串, to目的字符串  
函数原型:

```
char *strcpy(char *to, const char *from);
```

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

int main(int argc, const char * argv[]) {
    char str1[11] = "Hello";
    char str2[11];
    strcpy(str2, str1);
    //printf("%s\n",strcpy(str2, str1));
    printf("str2 = %s\n",str2);
    return 0;
}
```

程序运行结果如下:  
str2 = Hello

strcpy的函数的返回值为char\*类型, 主要用于形成函数链, 可直接使用返回值进行下一步操作。

## 课堂练习:

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

int main(int argc, const char * argv[]) {
    char string[80];
    strcpy( string, "Hello world from " );
    strcat( string, "strcpy " ); //字符串连接函数, 把第二个字符串
    参数连接到第一个字符串参数的结尾。
    strcat( string, "and " );
    strcat( string, "strcat!" );
    printf( "String = %s\n", string );
    return 0;
}
```

程序运行结果如下:

String = Hello world  
from strcpy and strcat!

- Function : strcmp 字符串比较，返回两个字符串比较后的ASCII码的差值。

函数原型：

```
int strcmp(const char *str1, const char *str2);
```

- 如果str1 > str2,返回值大于0;
- 如果str1 = str2,返回值等于0;
- 如果str1 < str2,返回值小于0;

两个字符串从前向后，逐个比较每对字符的ASCII码的大小；若相同，则继续比较，直到遇到第一对不同的字符。ASCII码整数大的字符为大。

```
NAME
    strcmp, strncmp -- compare strings

LIBRARY
    Standard C Library (libc, -lc)

SYNOPSIS
    #include <string.h>

    int
    strcmp(const char *s1, const char *s2);

    int
    strncmp(const char *s1, const char *s2, size_t n);

DESCRIPTION
    The strcmp() and strncmp() functions lexicographically compare the null-terminated strings s1 and s2.

    The strncmp() function compares not more than n characters. Because strncmp() is designed for comparing strings rather than binary data, characters that appear after a '\0' character are not compared.
```

## 课堂练习:

```
#include <stdio.h>
#include <string.h>
char string1[] = "The quick brown dog jumps over the lazy fox";
char string2[] = "The QUICK brown dog jumps over the lazy fox";
int main( void ){
    char tmp[20];
    int result;
    printf( "Compare strings:\n\t%s\n\t%s\n\n", string1, string2 );
    result = strcmp( string1, string2 );
    if( result > 0 )
        strcpy( tmp, "greater than" );
    else if( result < 0 )
        strcpy( tmp, "less than" );
    else
        strcpy( tmp, "equal to" );

    printf( "\tstrcmp:   String 1 is %s string 2\n", tmp );
    return 0;
}
```

程序运行结果如下:

Compare strings:

The quick brown dog jumps over the lazy fox

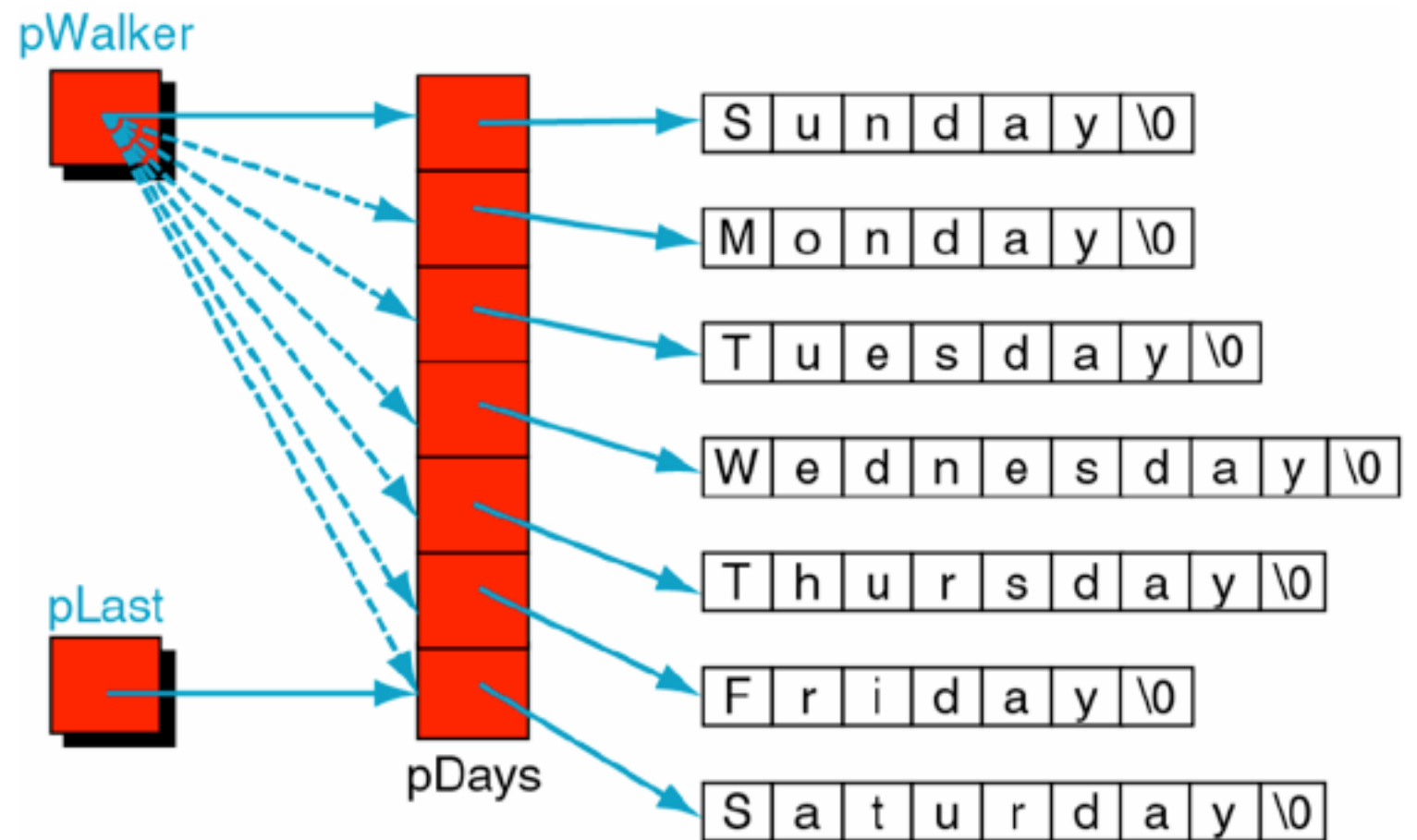
The QUICK brown dog jumps over the lazy fox

strcmp: String 1 is greater than string 2

## 四、字符串数组

字符串数组存储的，不是每一个字符串的内容，而是每个字符串首元素的地址。

```
char* pDays[7];  
pDays[0] = "Sunday";  
pDays[1] = "Monday";  
pDays[2] = "Tuesday";  
pDays[3] = "Wednesday";  
pDays[4] = "Thursday";  
pDays[6] = "Saturday";
```



字符串数组存储的，不是每一个字符串的内容，而是每个字符串首元素的地址。

```
char **pWalker, **pLast ; //字符串数组是一种二维指针。  
//数组名本身是指针，指向数组首元素；数组每个元素存储的也是指针，是每个字符串首字符的地址。  
pLast = &pDays[6]; //pLast指向字符串数组的最后一个元素  
for (pWalker = pDays; pWalker <= pLast; pWalker++)  
{  
    printf("%s\n", *pWalker );//*pWalker == pDays[i];  
}  
pWalker--;//此时pWalker指向pDays[6];  
printf("%p -> %s\n", pWalker, *pWalker);
```

程序输出结果：

```
Sunday  
Monday  
Tuesday  
Wednesday  
Thursday  
Friday  
Saturday  
0x7fff5fbff800 -> Saturday
```



iPhone



# The End