字符串函数之Strtok()函数

**Strtok()函数详解：**

该函数包含在  **"string.h"**头文件中   
**函数原型：**

1. char\* strtok (char\* str,constchar\* delimiters );

**函数功能：**   
切割字符串，将str切分成一个个子串     
**函数参数：**   
str：在第一次被调用的时间str是传入需要被切割字符串的首地址；在后面调用的时间传入NULL。     
delimiters：表示切割字符串（字符串中每个字符都会 当作分割符）。     
**函数返回值：**   
当s中的字符查找到末尾时，返回NULL;     
如果查不到delimiter所标示的字符，则返回当前strtok的字符串的指针。

**使用strtok()函数:**

1. [复制代码](javascript:void(0);)
2. #include<stdio.h>
3. #include<string.h>
4. int main(void)
5. {
6. char buf[]="hello@boy@this@is@heima";
7. char\*temp = strtok(buf,"@");
8. while(temp)
9. {
10. printf("%s ",temp);
11. temp = strtok(NULL,"@");
12. }
13. return0;

}

[复制代码](javascript:void(0);)

**预计输出结果：**

**"hello boy this is heima "**

**实际运行结果：**   
https://images2015.cnblogs.com/blog/749068/201703/749068-20170324120315986-1223370037.png

**得到的结论：**   
strtok在切割字符串的时间，实际上就是将分割符的字符delimiter替换为'\0'并且返回首地址。

**自己实现strtok()函数原理：**

**函数代码：**

1. [复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

#include<stdio.h>

#include<string.h>

//根据函数原型实现strtok()函数

char\* myStrtok\_origin(char\* str\_arr,constchar\* delimiters,char\*\*temp\_str)

{

//定义一个指针来指向待分解串

char\*b\_temp;

/\*

\* 1、判断参数str\_arr是否为空，如果是NULL就以传递进来的temp\_str作为起始位置；

\* 若不是NULL，则以str为起始位置开始切分。

\*/

if(str\_arr == NULL)

{

str\_arr =\*temp\_str;

}

//2、跳过待分解字符串

//扫描delimiters字符开始的所有分解符

str\_arr += strspn(str\_arr, delimiters);

//3、判断当前待分解的位置是否为'\0'，若是则返回NULL，否则继续

if(\*str\_arr =='\0')

{

return NULL;

}

/\*

\* 4、保存当前的待分解串的指针b\_temp，调用strpbrk()在b\_temp中找分解符，

\* 如果找不到，则将temp\_str赋值为待分解字符串末尾部'\0'的位置，

\* b\_temp没有发生变化；若找到则将分解符所在位置赋值为'\0',

\* b\_temp相当于被截断了，temp\_str指向分解符的下一位置。

\*/

b\_temp = str\_arr;

str\_arr = strpbrk(str\_arr, delimiters);

if(str\_arr == NULL)

{

\*temp\_str = strchr(b\_temp,'\0');

}

else

{

\*str\_arr ='\0';

\*temp\_str = str\_arr +1;

}

//5、函数最后部分无论找没找到分解符，都将b\_temp返回。

return b\_temp;

}

//使用myStrtok来简化myStrtok\_origin函数

char\* myStrtok(char\* str\_arr,constchar\* delimiters)

{

staticchar\*last;

return myStrtok\_origin(str\_arr, delimiters,&last);

}

int main(void)

{

char buf[]="hello@boy@this@is@heima";

//1、使用myStrtok\_origin()函数

char\*temp\_str = NULL;

char\*str = myStrtok\_origin(buf,"@",&temp\_str);

while(str)

{

printf("%s ",str);

str = myStrtok\_origin(NULL,"@",&temp\_str);

}

//2、使用myStrtok()函数

char\*str1 = myStrtok(buf,"@");

while(str1)

{

printf("%s ",str1);

str1 = myStrtok(NULL,"@");

}

return0;

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

**完美实现！！！**

**拓展知识点：可重入与不可重入函数**

**可重入函数：**   
指一个可以被多个任务调用的函数（过程），任务在调用时不必担心数据是否会出错。     
**不可重入函数：**   
如果函数接口的代码实现中在块内定义和使用了static存储类型的变量，这种函数将是不可重入函数。它在实时系统中是不安全函数