**strtok和strtok\_r函数总结**

1. Windows下的字符串分割函数strtok

（1）函数简介

函数原型：char \*strtok(char \*s, char \*delim);

函数功能：把字符串s按照字符串delim进行分割，然后返回分割的结果。

函数使用：

1.strtok函数的实质上的处理是，strtok在s中查找包含在delim中的字符并用NULL(’/0′)来替换,直到找遍整个字符串。

这句话有两层含义：

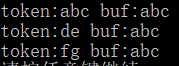
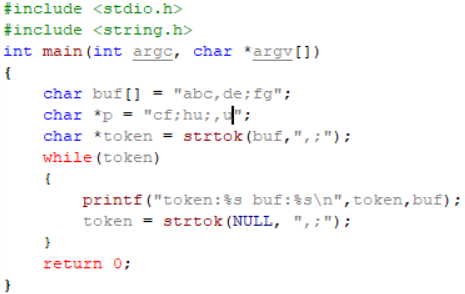
（1）每次调用strtok函数只能获得一个分割单位。

（2）要获得所有的分割单元必须反复调用strtok函数。

2.strtok函数第一次调用使用形参s，以后的调用时的需用NULL来替换s.

3.形参s(要分割的字符串)对应的变量应用char s[]=”….”形式，而不能用char \*s=”….”形式。

Code:



说明：

（1）分隔符可以是多个

（2）strtok函数会破坏被分解字符串的完整，调用前和调用后的buf已经不一样了。如果要保持原字符串的完整，可以使用strchr和sscanf的组合等。

（3）如果我们把token = strtok( NULL, ” “);换成token = strtok( buf, " ");则while循环会成为一个无限循环，而输出的结果就只为：abc。

（4）如果使用strtok(p,”，；”);会导致死机

为什么会死机呢，分析具体的原因，就涉及到strtok函数源码的实现：

在strtok函数体中有一个static char 类型的指针（假设为 char \*p）,它的作用就是令p=s,用于保存s的起始地址。在以后的strtok调用中用NULL来代替s的原因就是防止p被重新赋值而指向s的起始地址，从而可以保证p可以指向s的其他位置，直到最后分割完整个字符串。 但是如果用s来代替NULL，那么每次调用strtok时，p就会指向s的初始地址，从而只能获得第一个分割出来的字串，如上面的例子中while会是一个无限循环，而输出的结果只能是“abc”

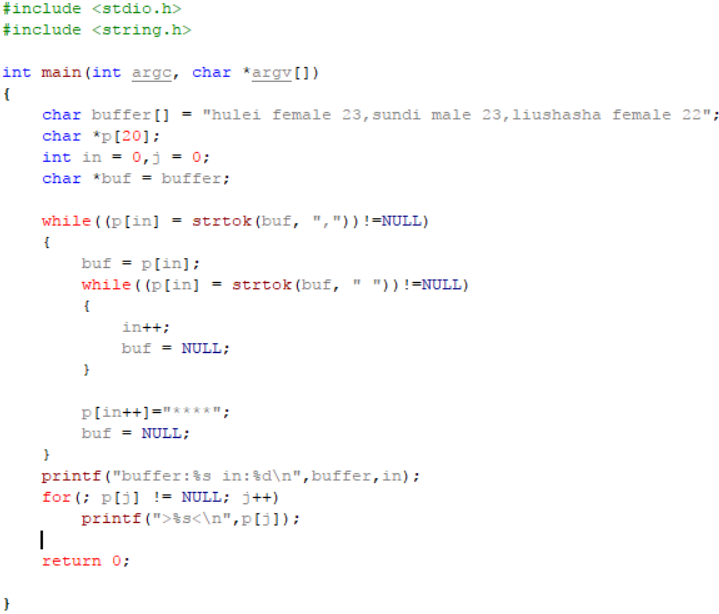
(2)strtok函数的缺陷

假如要将一个字符串“hulei female 23,sundi male 23,liushasha female 22”输出到一个结构体struct{char name[20],

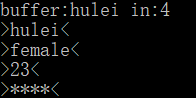
char sex[10],

int age};

code:



运行结果：



并没有像预期一样获取所有我们想要的字符串，所以得分析程序的运行过程是怎样的：

红色字体表示内部指针指向要修改字符串的位置，加粗字体表示修改内容

（1）“hulei female 23**’\0’**sundi male 23,liushasha female 22” //第一次外层循环，分隔符‘，‘

此时buf存储的字符串是“hulei female 23**’**\0**’**”，进入内层循环，分割符“ “

（2）“hulei**‘\0‘**female 23**’**\0**’**” //第一次内层循环，in=0

（3）“female**’\0’**23**’**\0**’**” //第二次内层循环，in=1

（4）“23‘\0‘“ //第三次内层循环，in=2，因为没有找到' '，内部指针返回指向23后的'/0'

（5）第四次内层循环，in=3，此时内部指针指向‘\0’无法提取，返回空值，结束内循环。

（6）“hulei female 23’\0’sundi male 23,liushasha female 22” //外部循环遇到‘\0’结束

因为指向所要查找的字符串一直都在改变，所以导致上面的结果，同时也可以看出这是一个线程不安全函数，Linux下就提出了一个线程安全函数strtok\_r

（二）linux线程安全函数strtok\_r

（1）函数简介：

函数原型：char \*strtok\_r(char \*str, const char \*delim, char \*\*saveptr);

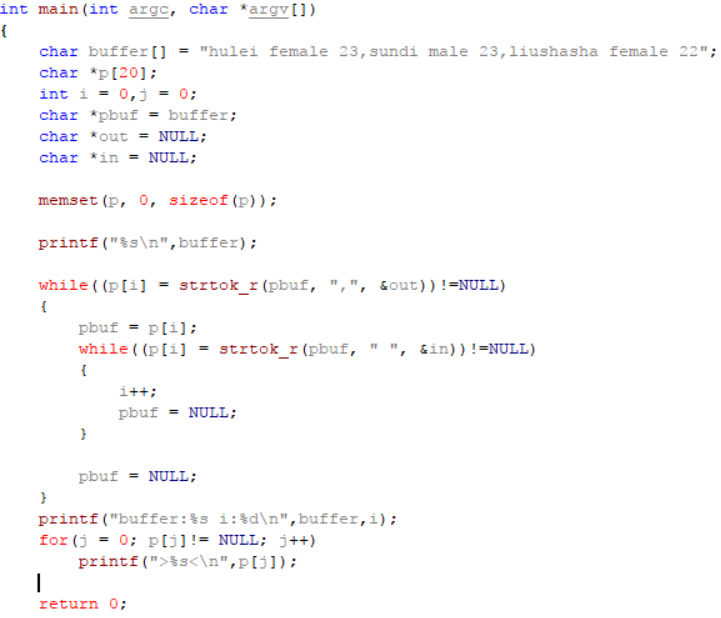
函数功能：把字符串s按照字符串delim进行分割，然后返回分割的结果。

函数说明：

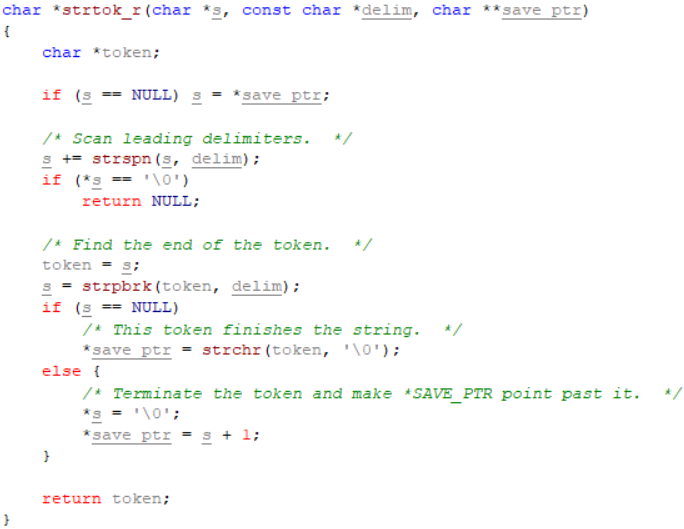
Linux下的线程安全函数

上面的例子，我们采用strtok\_r来写，外层循环提取个人全部信息，内层循环提取个人单项信息

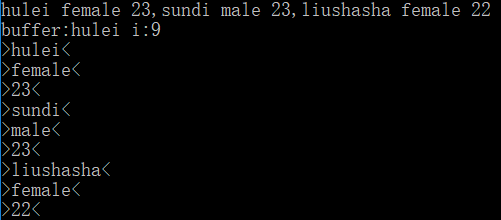
Code：



Strtok\_r源码（如果想在Windows下使用这个函数，直接将源码copy就行了）：



最终的输出结果：



程序的运行过程是这样的：

红色字体表示内部指针指向要修改字符串的位置，加粗字体表示修改内容

（1）“hulei female 23**’\0’**sundi male 23,liushasha female 22” //第一次外层循环，分隔符‘，‘

（此时外部循环的内部指针\*out指向字符‘s’,和内部循环的内部指针各自保存各自的）

此时buf存储的字符串是“hulei female 23**’**\0**’**”，进入内层循环，分割符“ “

（2）“hulei**‘\0‘**female 23**’**\0**’**” //第一次内层循环，in=0

（3）“female**’\0’**23**’**\0**’**” //第二次内层循环，in=1

（4）“23‘\0‘“ //第三次内层循环，in=2，因为没有找到' '，内部指针返回指向23后的'/0'

（5）第四次内层循环，in=3，此时内部指针指向‘\0’无法提取，返回空值，结束内循环。

（6）“hulei female 23’\0’sundi male 23,liushasha female 22” //第二次外部循环开始，分隔符‘，‘，strtok就是在这一步结束的。

（三）在VS2013下还有一个线程安全函数strtok\_s，