<https://blog.csdn.net/leichelle/article/details/7465769>

函数名: strdup

功  能: 将串拷贝到新建的位置处

用  法: char \*strdup(char \*str)；

这个函数在linux的man手册里解释为：

The strdup() function returns a pointer toa new string which is a

duplicate of the string s. Memory for thenew string is obtained with

malloc(3), and can be freed with free(3).

The strndup() function is similar, but onlycopies at most n charac-

ters. If s is longer than n, only ncharacters are copied, and a termi-

nating NUL is added.

strdup函数原型：

strdup()主要是拷贝字符串s的一个副本，由函数返回值返回，这个副本有自己的内存空间，和s不相干。strdup函数复制一个字符串，使用完后要记得删除在函数中动态申请的内存，strdup函数的参数不能为NULL，一旦为NULL，就会报段错误，因为该函数包括了strlen函数，而该函数参数不能是NULL。

strdup的工作原理:

char \* \_\_strdup (const char \*s)

{

size\_t len =strlen (s) + 1;

void \*new =malloc (len);

if (new == NULL)

return NULL;

return (char \*)memcpy (new, s, len);

}

实例1：

C/C++ code

#include <stdio.h>

#include <string.h>

#include <alloc.h>

int main(void)

{

char \*dup\_str,\*string = "abcde";

dup\_str =strdup(string);

      printf("%s\n", dup\_str);free(dup\_str); return 0;

}

实例2：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

unsigned int Test()

{

charbuf[]="Hello,World!";

char\* pb =strndup(buf,strlen(buf));

return (unsignedint)(pb);

}

int main()

{

unsigned int pch= Test();

printf("Testing:%s\n",(char\*)pch);

free((void\*)pch);

return 0;

}

在Test函数里使用strndup而出了Test函数仍可以操作这段内存，并且可以释放。

由这个问题而延伸出来的问题就是，如何让函数得到的内存数据传出函数但仍可用。

解决方法目前本人只想到两个： 一个外部变量，如传递一个内存块指针给函数，但这种做法就是你得传递足够的内存，也就是你不能事先知道这个函数到底要多大的BUFFER。

 另一种方法就是在函数内部申请static变量，当然这也是全局区的变量，但这种做法的缺点就是，当函数多次运行时，static变量里面的数据会被覆盖。这种类型的另一个方法就是使用全局变量，但这和使用static变量很相同，不同的是全局变量可以操作控制，而static变量如果不把它传出函数，就不可对它操作控制了。另一类方法就是上面所述的，利用堆里的内存来实现，但存在危险。strdup是从堆中分配空间的！strdup调用了malloc，所以它需要释放！对于堆栈：堆是由程序员来管理的，比如说new,malloc等等都是在堆上分配的！

栈是由编译器来管理的。