

专注于嵌入式

对所做的事情的理解越深，你就会做的越好。

- [目录视图](#)
- [摘要视图](#)
- [订阅](#)

[赠书 | 异步2周年.技术图书免费送](#) [每周荐书：渗透测试、K8s、架构（评论送书）](#) [项目管理+代码托管+文档协作，开发更流畅](#)

C语言的那些秘密之---函数返回局部变量

标签：[语言C编译器存储](#)

2011-08-17 13:06 27105人阅读 [评论\(18\)](#) [收藏](#) [举报](#)

分类：

[编程语言 \(2\)](#)

[作者同类文章X](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

一般的来说，函数是可以返回局部变量的。局部变量的作用域只在函数内部，在函数返回后，局部变量的内存已经释放了。因此，如果函数返回的是局部变量的值，不涉及地址，程序不会出错。但是如果返回的是局部变量的地址(指针)的话，程序运行后会出错。因为函数只是把指针复制后返回了，但是指针指向的内容已经被释放了，这样指针指向的内容就是不可预料的内容，调用就会出错。准确的说，**函数不能通过返回指向栈内存的指针(注意这里指的是栈，返回指向堆内存的指针是可以的)**。

下面以函数返回局部变量的指针举几个典型的例子来说明：

1：

```
[cpp]
01. #include <stdio.h>
02. char *returnStr()
03. {
04.     char *p="hello world!";
05.     return p;
06. }
07. int main()
08. {
09.     char *str;
10.     str=returnStr();
11.     printf("%s\n", str);
12.     return 0;
13. }
```

这个没有任何问题，因为"hello world!"是一个字符串常量，存放在**只读数据段**，把该字符串常量存放的**只读数据段**的首地址赋值给了指针，所以returnStr函数退出时，该字符串常量所在内存不会被回收，故能够通过指针顺利无误的访问。

2：

```
[html]
01. #include <stdio.h>
02. char *returnStr()
03. {
04.     char p[]="hello world!";
05.     return p;
06. }
07. int main()
08. {
09.     char *str;
10.     str=returnStr();
11.     printf("%s\n", str);
12.     return 0;
13. }
```

"hello world!"是**局部变量存放在栈中**。当returnStr函数退出时，栈要清空，局部变量的内存也被清空了，所以这时的函数返回的是一个已被释放的内存地址，所以有可能打印出来的是乱码。

3：

```
[html]
01. int func()
02. {
03.     int a;
04.     ....
05.     return a;    //允许
06. }
07.
08. int * func()
09. {
10.     int a;
11.     ....
12.     return &a;    //无意义，不应该这样做
13. }
```

关闭

局部变量也分局部自动变量和局部静态变量，由于a返回的是值，因此返回一个局部变量是可以的，无论自动还是静态，因为这时候返回的是这个局部变量的值，但不应该返回指向局部自动变量的指针，因为函数调用结束后该局部自动变量被抛弃，这个指针指向一个不再存在的对象，是无意义的。但可以返回指向局部静态变量的指针，因为静态变量的生存期从定义起到程序结束。

4：如果函数的返回值非要是一个局部变量的地址，那么该局部变量一定要申明为static类型。如下：

```
[html]
01. #include <stdio.h>
02. char *returnStr()
03. {
04.     static char p[]="hello world!";
05.     return p;
06. }
07. int main()
08. {
09.     char *str;
10.     str=returnStr();
11.     printf("%s\n", str);
12.
13.     return 0;
14. }
```

5：数组是不能作为函数的返回值的，原因是编译器把数组名认为是局部变量（数组）的地址。返回一个数组一般用返回指向这个数组的指针代替，而且这个指针不能指向一个自动数组，因为函数结束后自动数组被抛弃，但可以返回一个指向静态局部数组的指针，因为静态存储期是从对象定义到程序结束的。如下：

```
[html]
01. int* func( void )
02. {
03.     static int a[10];
04.     .....
05.     return a;
06. }
```

6：返回指向堆内存的指针是可以的

```
[html]
01. char *GetMemory3(int num)
02. {
03.     char *p = (char *)malloc(sizeof(char) * num);
04.     return p;
05. }
06. void Test3(void)
07. {
08.     char *str = NULL;
09.     str = GetMemory3(100);
10.     strcpy(str, "hello");
11.     cout<< str << endl;
12.     free(str);
13. }
```

程序在运行的时候用 malloc 申请任意多少的内存,程序员自己负责在何时用 free释放内存。动态内存的生存期由程序员自己决定,使用非常灵活。

顶

61

踩

2

-

-

- 上一篇[结合typedef更为直观的应用函数指针](#)
- 下一篇[linux文件系统启动流程---笔记整理](#)

相关文章推荐

- [浅谈C语言函数返回值--局部变量和局部变量地址](#)
- [【直播】计算机视觉原理及实战—屈教授](#)
- [C语言中函数的思考（可以返回局部变量吗）](#)
- [【套餐】深度学习入门视频课程—唐宇迪](#)
- [返回局部变量的问题](#)
- [【套餐】Hadoop生态系统零基础入门-侯勇蛟](#)
- [局部变量地址的返回](#)
- [【套餐】嵌入式Linux C编程基础--朱有鹏](#)

- [局部变量返回值](#)
- [【套餐】2017软考系统集成项目——任钰](#)
- [C语言--返回局部变量的地址](#)



关闭