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

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。准确的来说，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
#include <stdio.h>
char *returnStr()
{
    char *p="hello world!";
    return p;
}
int main()
{
    char *str;
    str=returnStr();
    printf("%s\n", str);
    return 0;
}
```

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
#include <stdio.h>
char *returnStr()
{
    char p[]="hello world!";
    return p;
}
int main()
{
    char *str;
    str=returnStr();
    printf("%s\n", str);
    return 0;
}
```

"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
int func()
{
      int a;
      ....
      return a;    //允许
}

int * func()
{
      int a;
      ....
      return &a;    //无意义，不应该这样做
}
```

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]

#include <stdio.h>
char *returnStr()
{
    static char p[]="hello world!";
    return p;
}
int main()
{
    char *str;
     str=returnStr();
    printf("%s\n", str);

    return 0;
}
```

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]

int* func( void )
{
    static int a[10];
    ........
    return a;
}
```

6：返回指向堆内存的指针是可以的

```
[html]

char *GetMemory3(int num)
{
char *p = (char *)malloc(sizeof(char) * num);
return p;
}
void Test3(void)
{
char *str = NULL;
str = GetMemory3(100);
strcpy(str, "hello");
cout<< str << endl;
free(str);
}
```

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。

- 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