程序1：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7083619)

1. #include <stdio.h>
2. **char** \*returnStr()
3. {
4. **char** \*p="hello world!";
5. return p;
6. }
7. **int** main()
8. {
9. **char** \*str;
10. str=returnStr();
11. printf("%s\n", str);
12. return 0;
13. }

#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，存放在静态数据区，把该字符串常量存放的静态数据区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，回为操作的一直是这个静态数据区，故能够通过指针顺利无误的访问。

程序2：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7083619)

1. #include <stdio.h>
2. **char** \*returnStr()
3. {
4. **char** p[]="hello world!";
5. return p;
6. }
7. **int** main()
8. {
9. **char** \*str;
10. str=returnStr();
11. printf("%s\n", str);
12. return 0;
13. }

#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!"是一个字符串常量，存放在静态数据区，没错，但是把一个字符串常量赋值给了一个局部变量(char []型数组)，该局部变量存放在栈中，相当于把静态数据赋值给局部变量后，操作的是这个局部变量。这是与前着最本质的区别，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，所以打印出来的是乱码。

程序3：  
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：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7083619)

1. #include <stdio.h>
2. **char** \*returnStr()
3. {
4. static **char** p[]="hello world!";
5. return p;
6. }
7. **int** main()
8. {
9. **char** \*str;
10. str=returnStr();
11. printf("%s\n", str);
12. return 0;
13. }

#include <stdio.h>

char \*returnStr()

{

static char p[]="hello world!";

return p;

}

int main()

{

char \*str;

str=returnStr();

printf("%s\n", str);

return 0;

}

程序4：  
这个问题可以通过下面的一个例子来更好的说明：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7083619)

1. #include <stdio.h>
2. //返回的是局部变量的地址，该地址位于动态数据区，栈里
3. **char** \*s1()
4. {
5. **char** p[]="Hello world!";
6. printf("in s1 p=%p\n", p);
7. printf("in s1: string's address: %p\n", &("Hello world!"));
8. return p;
9. }
10. //返回的是字符串常量的地址，该地址位于静态数据区
11. **char** \*s2()
12. {
13. **char** \*q="Hello world!";
14. printf("in s2 q=%p\n", q);
15. printf("in s2: string's address: %p\n", &("Hello world!"));
16. return q;
17. }
18. //返回的是静态局部变量的地址，该地址位于静态数据区
19. **char** \*s3()
20. {
21. static **char** r[]="Hello world!";
22. printf("in s3 r=%p\n", r);
23. printf("in s3: string's address: %p\n", &("Hello world!"));
24. return r;
25. }
26. **int** main()
27. {
28. **char** \*t1, \*t2, \*t3;
29. t1=s1();
30. t2=s2();
31. t3=s3();
32. printf("in main:");
33. printf("p=%p, q=%p, r=%p\n", t1, t2, t3);
34. printf("%s\n", t1);
35. printf("%s\n", t2);
36. printf("%s\n", t3);
37. return 0;
38. }
39. 运行输出结果：
40. in s1 p=0xbff92efb
41. in s1: string's address: 0x80486ac
42. in s2 q=0x80486ac
43. in s2: string's address: 0x80486ac
44. in s3 r=0x804998c
45. in s3: string

#include <stdio.h>

//返回的是局部变量的地址，该地址位于动态数据区，栈里

char \*s1()

{

char p[]="Hello world!";

printf("in s1 p=%p\n", p);

printf("in s1: string's address: %p\n", &("Hello world!"));

return p;

}

//返回的是字符串常量的地址，该地址位于静态数据区

char \*s2()

{

char \*q="Hello world!";

printf("in s2 q=%p\n", q);

printf("in s2: string's address: %p\n", &("Hello world!"));

return q;

}

//返回的是静态局部变量的地址，该地址位于静态数据区

char \*s3()

{

static char r[]="Hello world!";

printf("in s3 r=%p\n", r);

printf("in s3: string's address: %p\n", &("Hello world!"));

return r;

}

int main()

{

char \*t1, \*t2, \*t3;

t1=s1();

t2=s2();

t3=s3();

printf("in main:");

printf("p=%p, q=%p, r=%p\n", t1, t2, t3);

printf("%s\n", t1);

printf("%s\n", t2);

printf("%s\n", t3);

return 0;

}

运行输出结果：

in s1 p=0xbff92efb

in s1: string's address: 0x80486ac

in s2 q=0x80486ac

in s2: string's address: 0x80486ac

in s3 r=0x804998c

in s3: string

可以看出，s1中两地址不同，s2中两地址相同，s3中两地址也不同  
  
这个结果正好应证了上面解释，同时，还可是得出一个结论：字符串常量，之所以称之为常量，因为它可一看作是一个没有命名的字符串且为常量，存放在静态数据区。这里说的静态数据区，是相对于堆、栈等动态数据区而言的。静态数据区存放的是全局变量和静态变量，从这一点上来说，字符串常量又可以称之为一个无名的静态变量，因为"Hello world!"这个字符串在函数 s1和s2 中都引用了，但在内存中却只有一份拷贝，这与静态变量性质相当神似。

程序5：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/lwbeyond/article/details/7083619)

1. **int** fn()
2. {
3. **int** j ＝ 3;
4. return j;
5. }
6. **int** main(**int** argc, **char**\* argv[])
7. {
8. printf("%d\n", fn());
9. return 0;
10. }

int fn()

{

int j ＝ 3;

return j;

}

int main(int argc, char\* argv[])

{

printf("%d\n", fn());

return 0;

}

首先要明白函数是怎么返回的，函数返回时,先将返回值存入寄存器,然后释放局部变量空间,最后return. fn()返回j其实是返回j的一个拷贝出来。