**1. char\* 和char[]的区别**

1. char[] p表示p是一个数组指针，相当于const pointer,不允许对该指针进行修改。但该指针所指向的数组内容，是分配在栈上面的，是可以修改的。

2. char \* pp表示pp是一个可变指针，允许对其进行修改，即可以指向其他地方，如pp = p也是可以的。对于\*pp = "abc";这样的情况，由于编译器优化，一般都会将abc存放在常量区域内，然后pp指针是局部变量，存放在栈中，因此，在函数返回中，允许返回该地址（实际上指向一个常量地址，字符串常量区）；而，char[] p是局部变量，当函数结束，存在栈中的数组内容均被销毁，因此返回p地址是不允许的。

同时，从上面的例子可以看出，cout确实存在一些规律：

1、对于数字指针如int \*p=new int; 那么cout<<p只会输出这个指针的值，而不会输出这个指针指向的内容。  
2、对于字符指针入char \*p="sdf f";那么cout<<p就会输出指针指向的数据，即sdf f

那么，像&(p+1)，由于p+1指向的是一个地址，不是一个指针，无法进行取址操作。

&p[1] = &p + 1，这样取到的实际上是从p+1开始的字符串内容。

分析上面的程序：

\*pp = "abc";

p[] = "abc";

\*pp指向的是字符串中的第一个字符。

cout << pp; // 返回pp地址开始的字符串：abc

cout << p; // 返回p地址开始的字符串：abc

cout << \*p; // 返回第一个字符：a

cout << \*(p+1); // 返回第二个字符：b

cout << &p[1];// 返回从第二个字符开始的字符串：bc

**2.** 预备知识—程序的内存分配

一个由c/C++编译的程序占用的内存分为以下几个部分  
1、栈区（stack）—由编译器自动分配释放，存放函数的参数值，局部变量的值等。其操作方式类似于数据结构中的栈。  
2、堆区（heap）—一般由程序员分配释放，若程序员不释放，程序结束时可能由OS回收。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表，呵呵。  
3、全局区（静态区）（static）—全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。程序结束后由系统释放。  
4、文字常量区—常量字符串就是放在这里的。程序结束后由系统释放。  
5、程序代码区

[**http://blog.csdn.net/yahohi/article/details/7427724**](http://blog.csdn.net/yahohi/article/details/7427724)