区别：

memcpy和memmove（）都是C语言中的库函数，在头文件string.h中，作用是拷贝一定长度的内存的内容，原型分别如下：

1. **void** \*memcpy(**void** \*dst, **const** **void** \*src, size\_t count);
2. **void** \*memmove(**void** \*dst, **const** **void** \*src, size\_t count);

他们的作用是一样的，唯一的区别是，当内存发生局部重叠的时候，memmove保证拷贝的结果是正确的，memcpy不保证拷贝的结果的正确。

一、memcpy函数

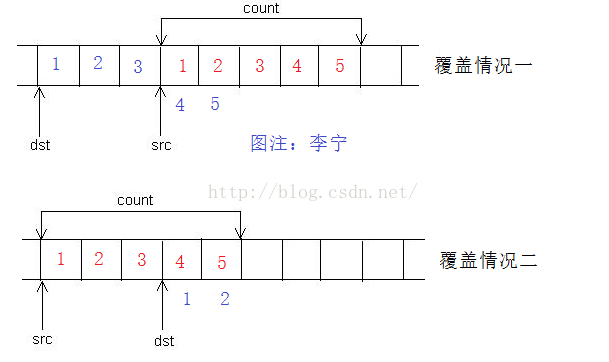
Memcpy原型：

1. **void** \*memcpy(**void** \*dest, **const** **void** \*src, size\_t n);

描述：  
        memcpy()函数从src内存中拷贝n个字节到dest内存区域，但是源和目的的内存区域不能重叠。  
返回值：  
        memcpy()函数返回指向dest的指针。  
二、memmove函数

memmovey原型：

1. **void** \*memmove(**void** \*dest, **const** **void** \*src, size\_t n);

描述：  
       memmove() 函数从src内存中拷贝n个字节到dest内存区域，但是源和目的的内存可以重叠。  
返回值：  
        memmove函数返回一个指向dest的指针。  
  
从上面的描述中可以看出两者的唯一区别就是在对待重叠区域的时候，memmove可以正确的完成对应的拷贝，而memcpy不能。  
  
内存覆盖的情形有以下两种，  


先看memcpy()和memmove()这两个函数的实现：

1. **void**\* my\_memcpy(**void**\* dst, **const** **void**\* src, size\_t n)
2. {
3. char \*tmp = (char\*)dst;
4. char \*s\_src = (char\*)src;
6. **while**(n--) {
7. \*tmp++ = \*s\_src++;
8. }
9. **return** dst;
10. }

从实现中可以看出memcpy()是从内存左侧一个字节一个字节地将src中的内容拷贝到dest的内存中，这种实现方式导致了对于图中第二种内存重叠情形下，最后两个字节的拷贝值明显不是原先的值了，新的值是变成了src的最开始的2个字节了。  
  
而对于第一种内存覆盖情况，memcpy的这种拷贝方式是可以的。

而memmove就是针对第二种内存覆盖情形，对memcpy进行了改进，改进代码如下：

1. **void**\* my\_memmove(**void**\* dst, **const** **void**\* src, size\_t n)
2. {
3. char\* s\_dst;
4. char\* s\_src;
5. s\_dst = (char\*)dst;
6. s\_src = (char\*)src;
7. **if**(s\_dst>s\_src && (s\_src+n>s\_dst)) {      //-------------------------第二种内存覆盖的情形。
8. s\_dst = s\_dst+n-1;
9. s\_src = s\_src+n-1;
10. **while**(n--) {
11. \*s\_dst-- = \*s\_src--;
12. }
13. }**else** {
14. **while**(n--) {
15. \*s\_dst++ = \*s\_src++;
16. }
17. }
18. **return** dst;
19. }

在第二种内存覆盖的情形下面，memcpy会出错，但是memmove是能正常工作的。