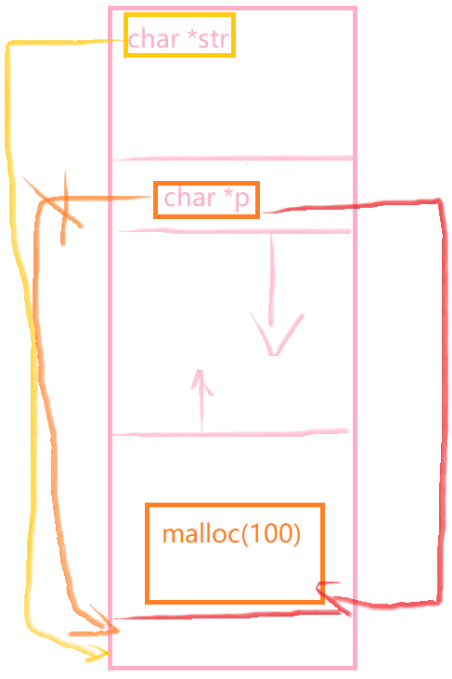
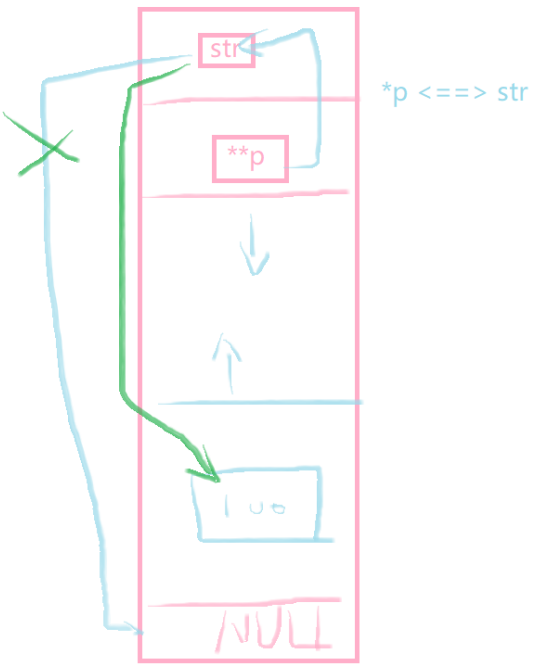
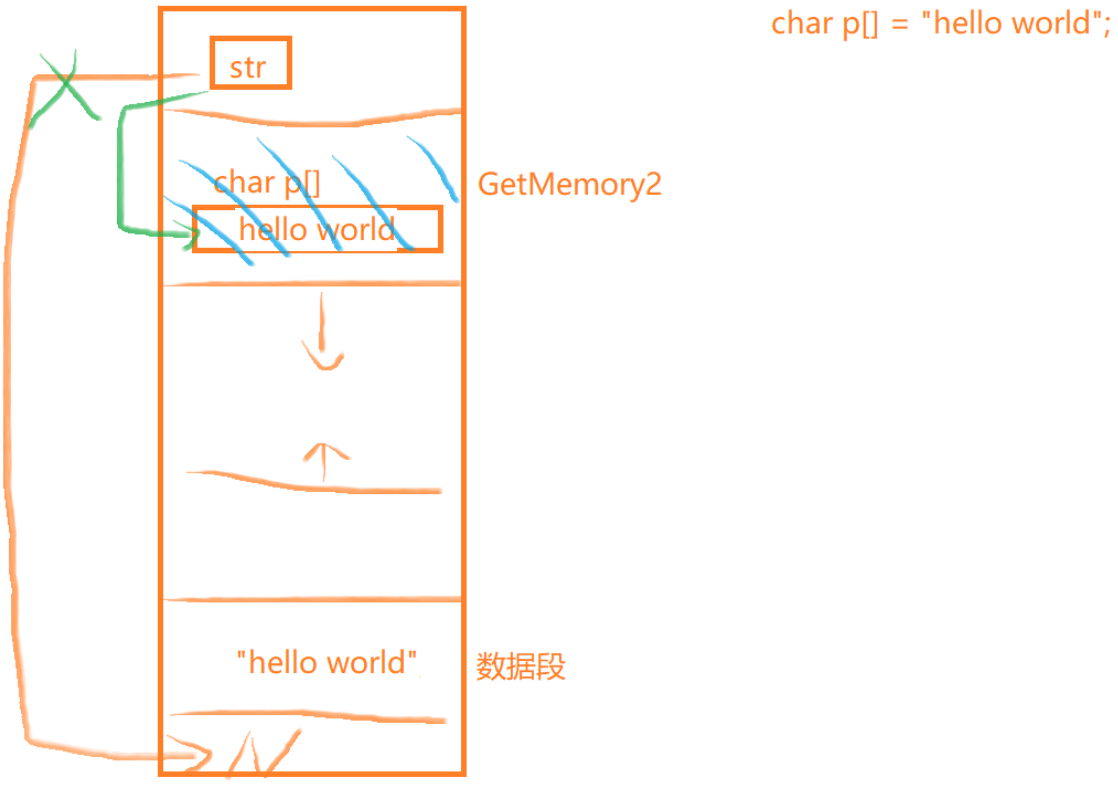
```c
// TEST1
            // 把p修改为二级指针， 用来接收 指针str的地址
void GetMemory1(char **p) // p ==== NULL
{
    *p = (char *)malloc(100); // 向系统申请100个字节的堆空间， 让p 指向该区域
}

void Test1(void)
{
    char *str = NULL;
    GetMemory1(&str);  /// str ==== NULL
    // 经过GetMemory1 的操作后 str的指向依然没有变换还是指向NULL
    strcpy(str, "hello world");  // 提示： 拷贝字符串， 把"hello world" 拷贝到str 所指向的内存空间中
    // 拷贝函数出现段错误

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






```c
// TEST2
char *GetMemory2(void)
{
```

```
    // 数组 p 所存放的位置为 栈空间， 当函数 GetMemory2 退出返回时， 该区域会被系统回收
    // 不应该返回该内存中的地址
    // 可以使用 static 来修饰该数组， 使其的内存区域改为数据段
    char p[] = "hello world";
    return p;
}
void Test2(void)
{
    char *str = NULL;
    str = GetMemory2();
    printf("TEST-2:%s\n",str);
}
```




```
// TEST3
char *GetMemory3(void)
{
    // 直接返回 常量区的内存地址 ， 注意该区域只读
    return "hello world";
}
void Test3(void)
```

```
{
    char *str = NULL;
     str = GetMemory3();
     printf("TEST-3:%s\n",str);
}
```

```
// TEST4 就是第一个的改版
void GetMemory4(char **p, int num)
{
    *p = (char *)malloc(num);
}
void Test4(void)
{
    char *str = NULL;
    GetMemory4(&str, 100);
     strcpy(str, "hello");
     printf("TEST-04:%s\n",str);
     free(str);
}
```

```

// TEST5
void Test5(void)
{
    char *str = (char *) malloc(100);
    strcpy(str, "hello");
    free(str);  // 把堆空间进行释放， 但是str依然指向堆空间的位置
    if(str != NULL)
    {
        strcpy(str, "world");  // 虽然可以拷贝， 但是属于非法访问
        printf("TEST-5:%s\n",str);
     }
}

```

```
1
2  // TEST6
3  void Test6()
4  {
5      char *str=(char *)malloc(100);
6      strcpy(str, "hello");
7      str+=6; //
8
9      if(str!=NULL)
10     {
11         strcpy(str, "world");
12         printf("TEST-6:%s\n" ,str); // 输出 wrold
13         printf("TEST-6:%s\n" ,str-=6); // 输出 Hello
14
15     }
16
17     free(str); // 释放的时候必须使用 最初申请得到的那个首地址（入口地址）
18 }
19
```