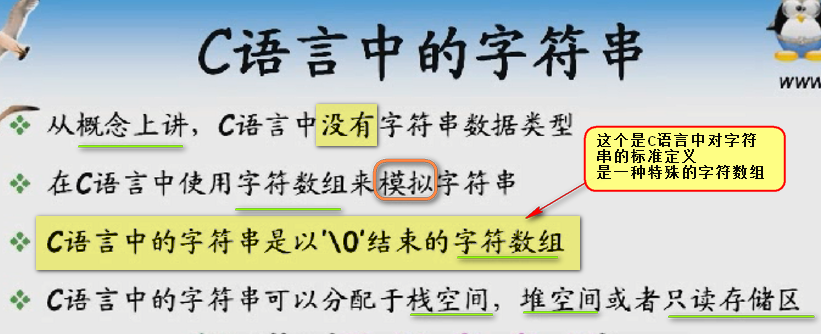
C语言中的字符串



C语言概念上没有字符串类型 所以 只能用字符数组模拟字符串



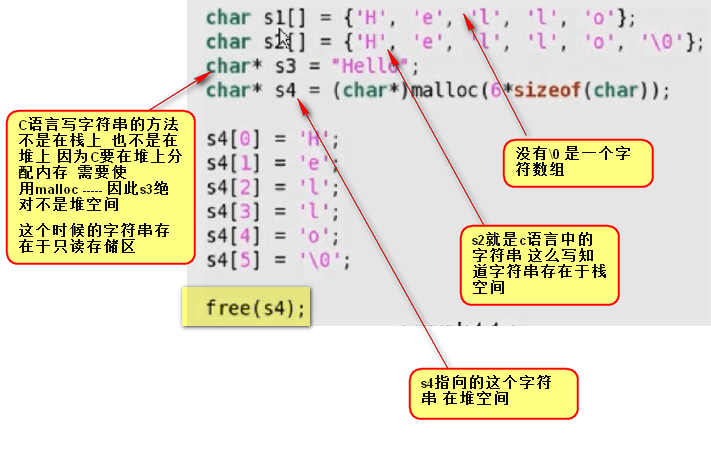
**如果说 一个字符数组 没有\0 是字符串么？ 不是！**

C语言中的字符串可以分配与栈空间 堆空间和只读存储区

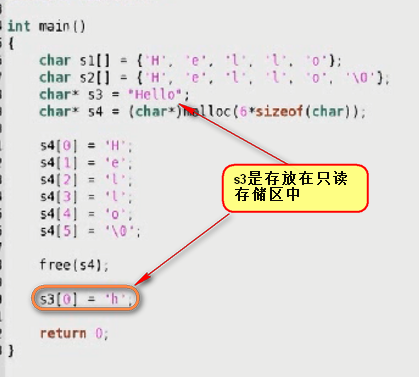
**---- 如何理解字符串存在于程序的只读存储区？**

 这个s3存在于只读存储区 那么 我的s3就不能改变指向的这个字符串中的内容 ---- 因为是只读。。。---- 所以 这里面的意思就是 Hello不能被修改为hello这样的

这样 程序运行错误

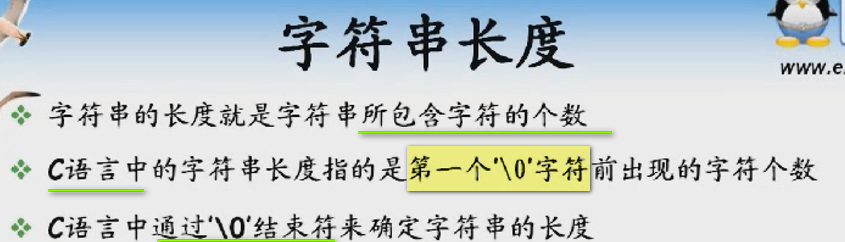


看看只读存储区中的数据能不能被改变？

 编译可以通过 但是 运行

 告诉程序爆了 -- 段错误 --- 因为s3指向了只读存储区的 只读的存储区 要去改变内容 就会差生错误 ---- 就会被OS立即结束掉 --- OS检测掉这个行为 立刻被结束掉

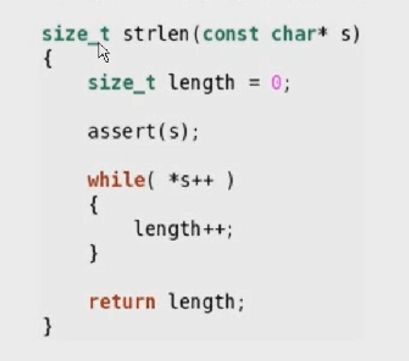
栈空间分配的字符串 和堆空间中分配的字符串 都可以自由的改动里面的任意一个字符 但是 只读存储区中的字符串 不能进行任何改动 ===== 所以 这个”Hello”就是一个字符串常量 ---- 一旦改变 运行出现段错误



【和Java中计算字符串的长度不一样】

假设有一个字符数组 有100个元素 第十个元素是\0 那么 当成字符串来看 就只有10（0-9）

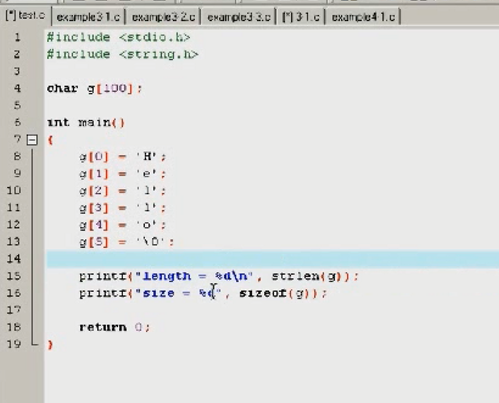
strlen这个就是计算C语言中字符串的长度

 这个是strlen的库函数的一种实现

【参数是const char\* s const出现在\*的左边 左值右指 ---- 所以 是表示这个字符串的内容是常量的意思】

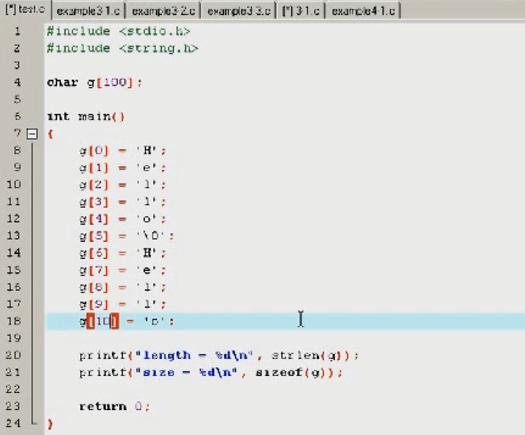
【所以 C语言中 判断到 \0的时候 就是条件就是假 这样length就不再计数 返回字符串的长度】

assert(s);判断s是否为空



 可以看到字符数组中 代表字符串的长度只有5 但是 整个g的大小是100个字节

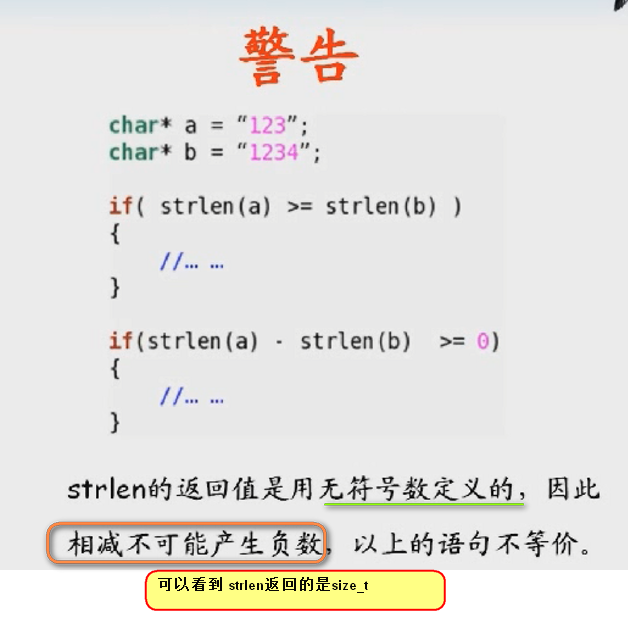
C语言 字符串是字符数组来实现的 必须有\0



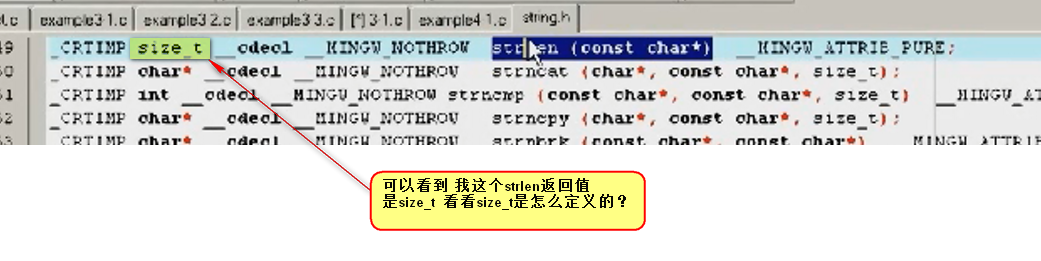
 运行结果还是没有变化

我们的字符串 都是第一个\0之前的所有内容 后面都不是字符串的一部分

**这就很好的展现了 字符串和字符数组不一样的地方**



看下strlen返回值如何定义的

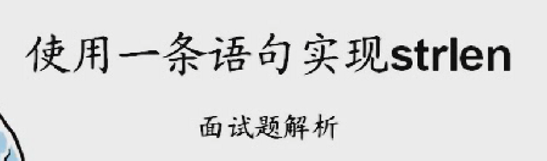


size\_t是一个无符号数 两个无符号数相减 还是无符号数 ---- 无符号数相减总是>0

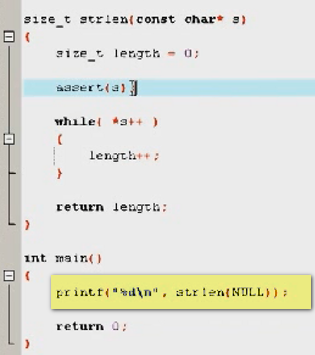
这样上下两个if语句不等价

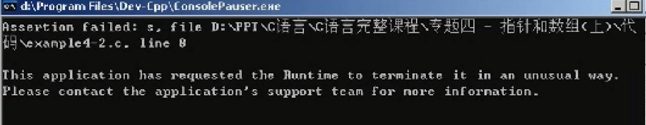
所以 下面的if总是执行

所以 必须知道strlen返回是unsigned类型 一定要知道不要写出类似的语句了

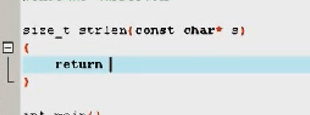


之前那个是使用循环实现的



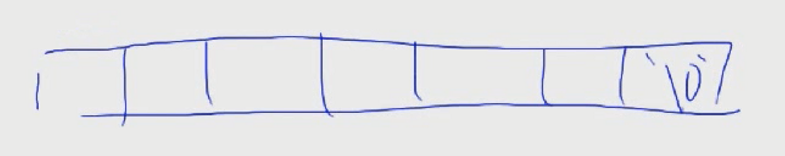


看结束符之前有几个字符 ---- 使用递归

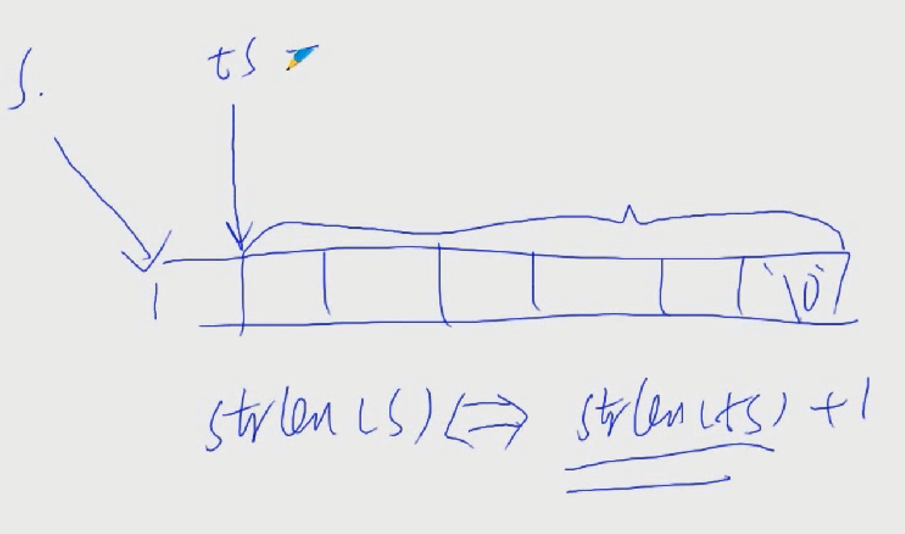


有一个实现 ---- 就是判断是否到了反斜杠0 ---- 一条语句如何判断 ---- 三目运算符

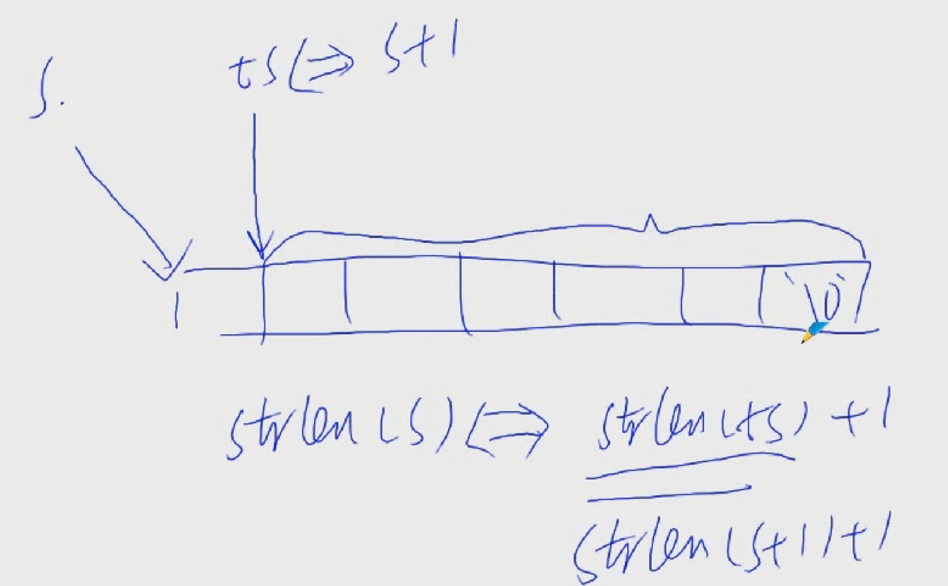
 因为之前循环的版本 是一个个数数 现在只能是一条语句 用递归代替循环



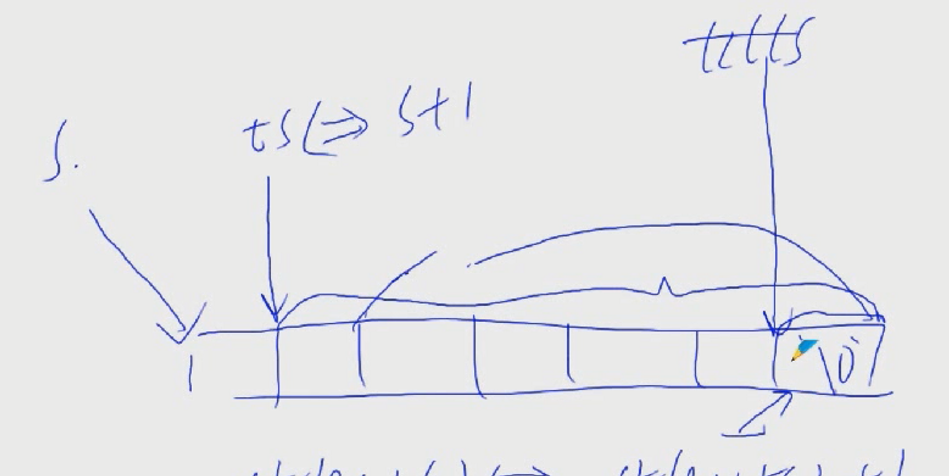
递归来求s的长度 等价于先求一下前一部分的长度 就比现在的长度小1

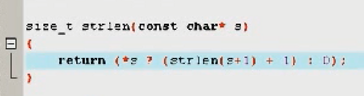


根据指针运算法则 这个strlen(ts)就是s+1



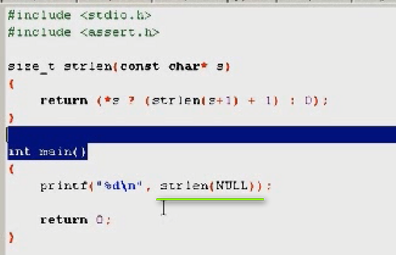
推断到 ----

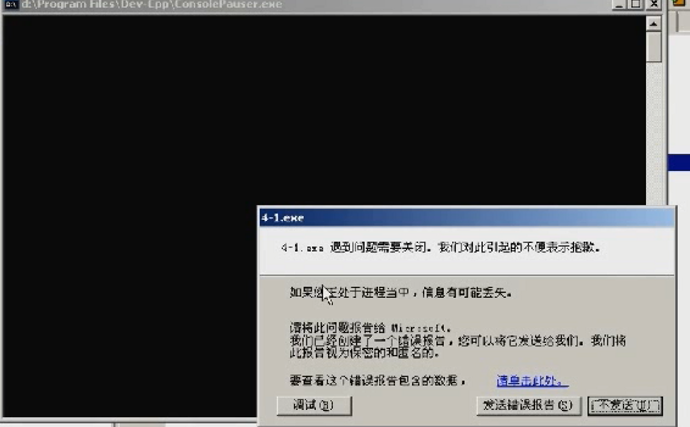
 当这个指针tttts指向这里的时候 说明子串的长度此时是0 --- 我们的代码已经给出来了 





一行代码完成 做出来 但是 没有处理NULL

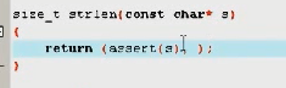
 运行的时候



如何加入assert语句？

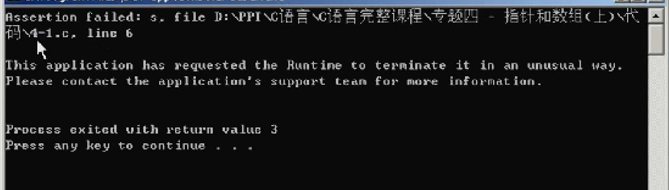
已经是三目运算符 在哪里加？ 没关系 我们还有逗号表达式

逗号表达式放在最后一个 之前的值都可以忽略不计

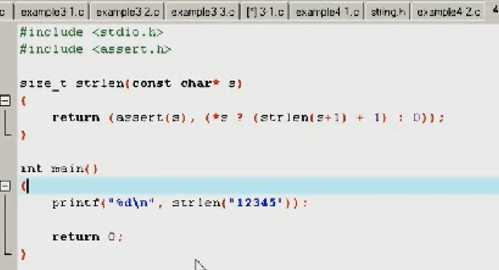




\*s? 就是判断是否是\0



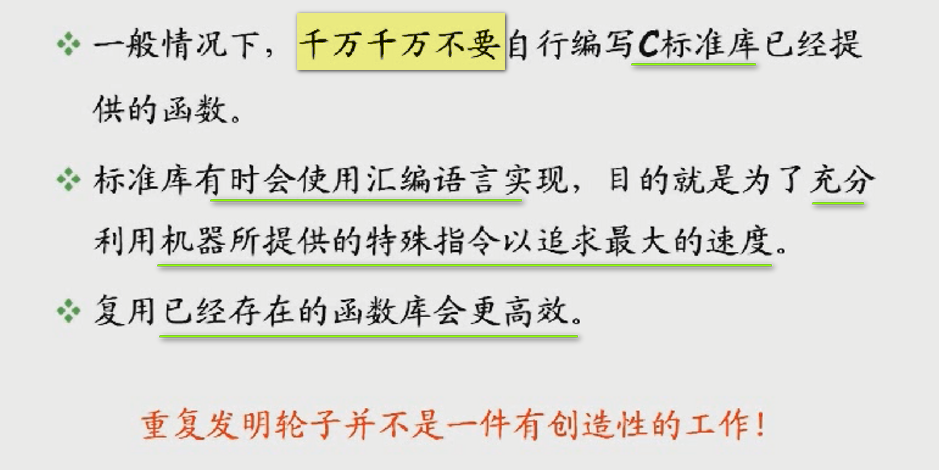
异常给出来了



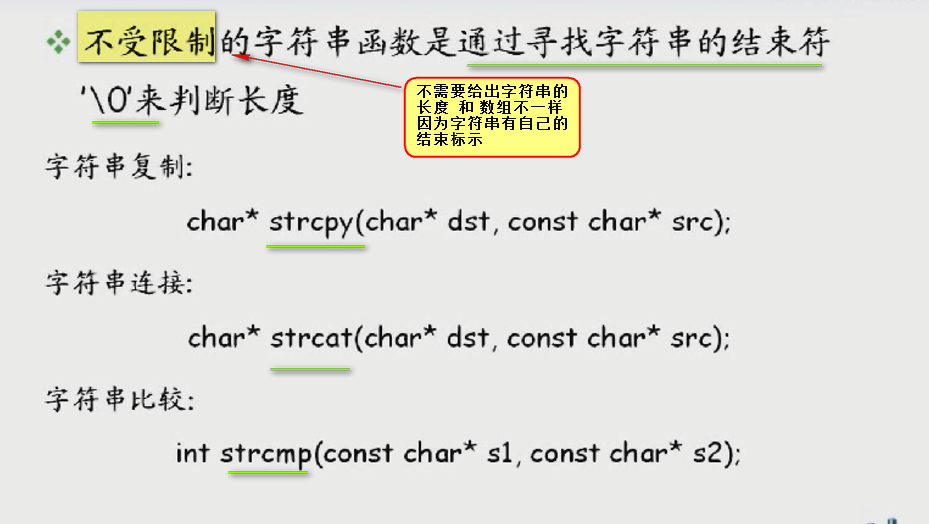
这样写之后呢 一行代码就把strlen一行语句给实现了

Return里面一个是逗号表达式 另一个是三目运算符表达式 ---- 求长度的表达式

【循环和递归的关系】



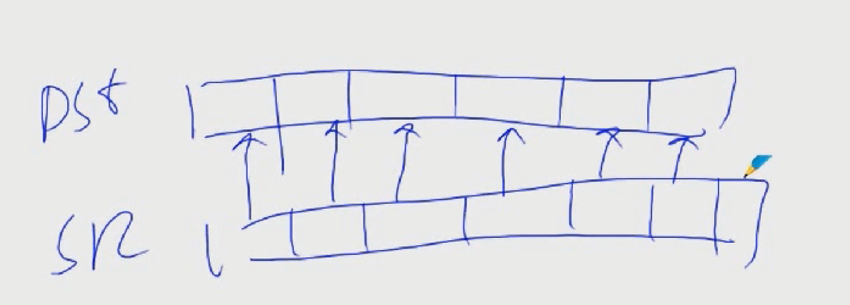
公司招聘人的时候 通过一系列手段 看看是否符合 ----- 看看你能否灵活运用逗号表达式 三目运算符 是否会递归



如果我们的参数 是一个字符数组 而不是字符串 导致我们这些函数调用出错 因为这些函数会找\0 ---- 调用失败和出错

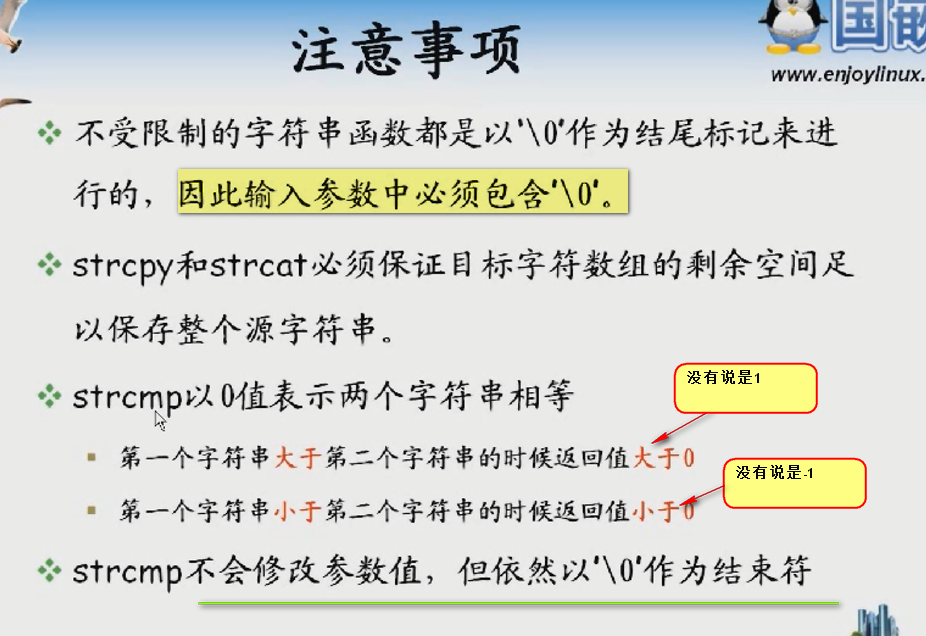
还有strcpy和strcat必须要保证目标里面有足够容纳后面的字符串的空间 如果不够 也会出错 -----

strcpy的工作方式

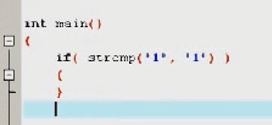


Dst空间不够了 后面的空间 src比较大的时候 dst比较小 函数不会检测 使用了不是我们目标空间 就会出错

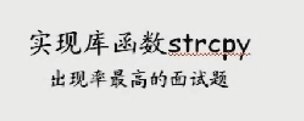
对于strcat 字符串连接 同样存在这个问题



如果使用strcmp

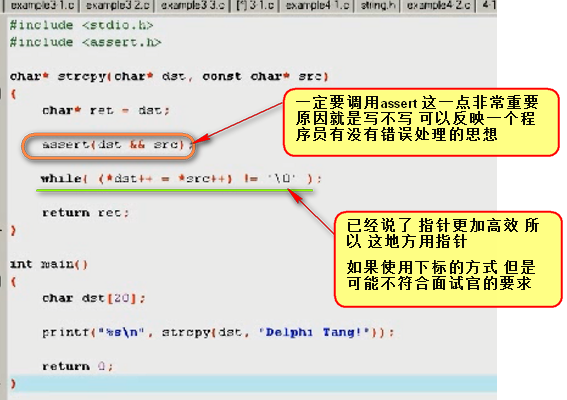
  
这样的代码写出来 就是 如果第一个字符串和第二个字符串相等 --- 返回0 所以 这样if不执行

 要和==0做比较



直接看代码

Strcpy的返回值 是目标本身 所以 程序的第一行 就要指定一个指针



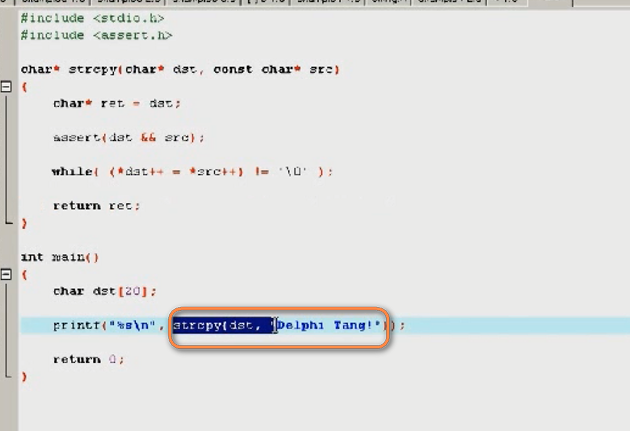
四条语句 都非常重要

也就是 第一条语句

char\* ret = dst; ---- 考察会不会链式法则 ----- 比如 我们的printf需要的参数正好是strcpy的结果 ----

第二条反应有没有安全编程策略

第三条反应我们的程序员是否知道指针 递增的方式比下标的方式效率高



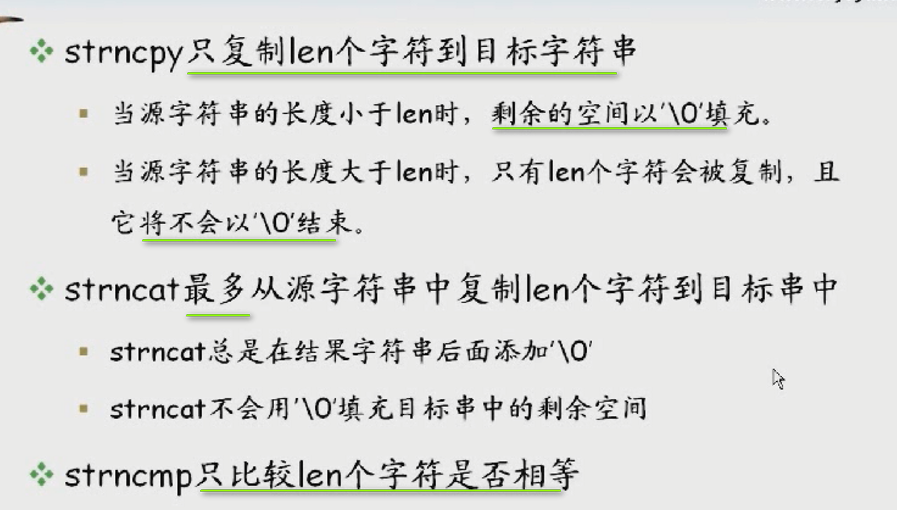
 打印出了原来的字符串

C语言中另一组函数

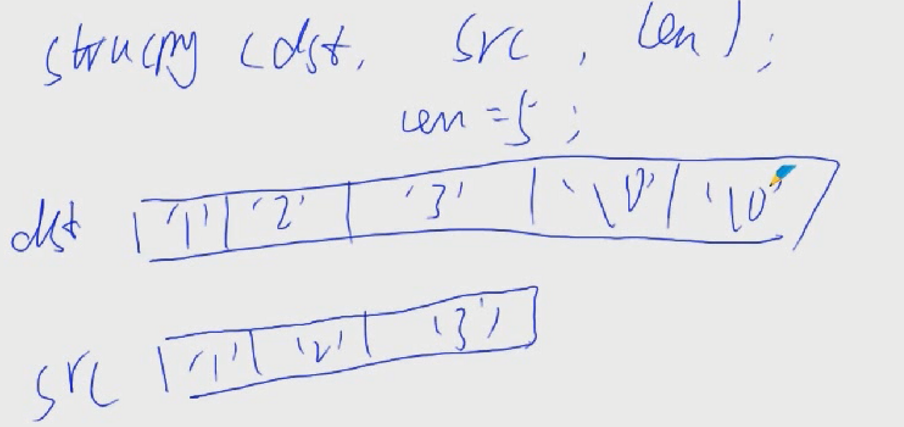


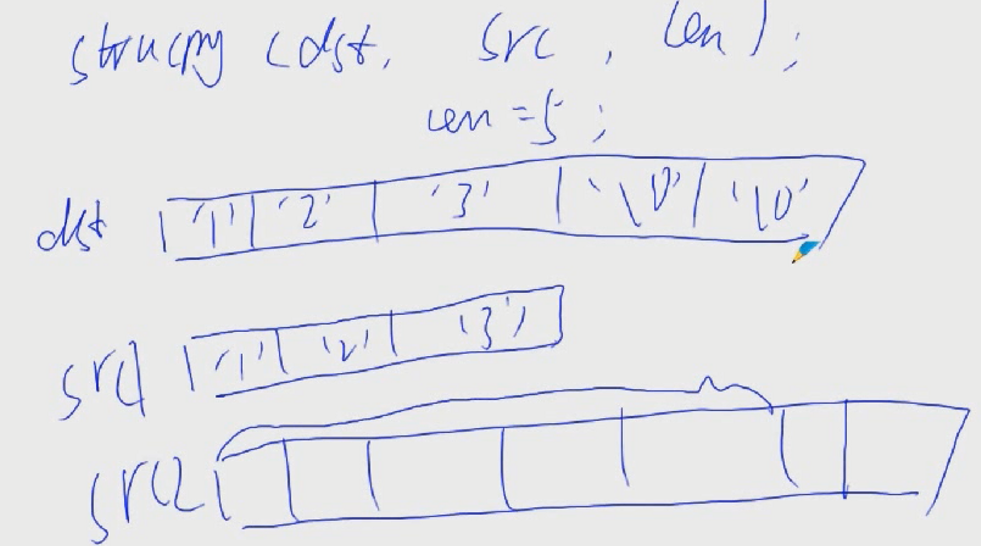
限定于多少个字符之内 ---- 多了一个参数

如果非要调用字符串函数 库函数 一定要调用长度受限的函数 ---手工指定！！！---- 比不受限制的安全得多！

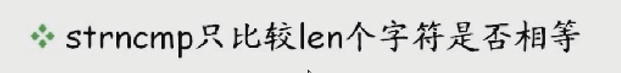


strncpy 我们的dst目标





 表示结束 但是 不会填充剩余的空间

 0在C语言中表示false

===================== char\* p = “Hello World”; 定义方式理解 ======================

可以看出来 char\* p = “Hello World”;

这样的字符串指针 实际定义的类型是指向一个char的指针 而不是指向整个字符串 ----- 这是从含义上面这么说 ----- 但是 C语言中每一个字符串的结尾都是’\0’ ----- 所以 我其实是给了指向这个字符串的首元素的地址 比如H的地址 这样 就是

char h = ‘H’;

char\* p = &h;

或者 严谨一点：

char strHW[12] = {‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘ ’, ‘W’, ‘o’, ‘r’, ‘l’, ‘d’, ‘\0’};

char\* p = &strHW[0]; 或者char\* p = strHW ---- 因为数组名是数组首元素的地址 ------ 但是 这样定义是严谨 但是非常麻烦 --------- 因此 为了简洁：char\* p=”Hello World”; 一劳永逸 这个”Hello World”里面就含有了\0 ---- 这样 我实际上是给了这个字符串数组（因为在常量区分配的 所以 不准确）的首元素的地址 赋给了p 而不是整个数组的地址赋给了p ----- 就给了一个开头 然后结尾通过判断是否是\0 来进行终结 ----- 所以 char\* p=”Hello World”;实际上是隐含给了字符数组的首元素的地址+隐含的字符串终结字符 ---- 这样 有了首位置 和 尾部标志 就能确定一个字符串本身了

但是 从字符数组 char strHW[12] = {‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘ ’, ‘W’, ‘o’, ‘r’, ‘l’, ‘d’, ‘\0’}; 来看 ---- strHW是字符数组首元素的地址 ---- 所以 shrHW的类型是char\* ---- 整个字符数组的地址应该是&strHW ---- 对应的指针的类型应该是char(\*pArray)[12] ------ 也就是指向数组的指针

可以定义为 typedef char(\*PArrayT)[12];

PArrayT pArray = &strHW;

所以 千万不要误以为 char\* p = “Hello World”; p就变成了指向数组的指针 这个是错误的理解！！！！！！ 仅仅为了方便才这样定义 实际上 p是指向数组首元素的 ----- 所以 是char\* 而不是char(\*p)[12] 类型根本就是不一样的！