1. strlen, strcpy, sizeof

对strlen的掌握，它没有包括字符串末尾的'\0'

sizeof是实际占用的内存空间，对于字符串数组，它的大小是实际的字符串长度再加’\0’的长度；而对于指针（包括字符串常量指针），它的大小应该是4（对于32位操作系统）

2. 自己模拟编写strlen，strcpy

|  |
| --- |
| int strlen( const char \*str ) //输入参数const  {  assert( strt != NULL ); //断言字符串地址非0  　int len;  　while( (\*str++) != '\0' ) {  　　len++;  　}  　return len;  } |
| char \* strcpy( char \*strDest, const char \*strSrc ) { //输入参数const保护起来  　assert( (strDest != NULL) && (strSrc != NULL) ); //保证传递的地址非NULL  　char \*address = strDest;    　while( (\*strDest++ = \* strSrc++) != ‘\0’ ); //注意括号的优先级，而且最终的一个’\0’还是被拷贝了的  　　return address; //返回原先的地址  } |

3. const char \*pstr1 和 char\* const pstr2 的区别

const char \*pstr1; pstr1是常量指针，pstr1指向的地址内容不能被改变，但是pstr1可以指向别的地址，比如

不能使用pstr1[3]=’A’;但是可以pstr1=adder\_2;来指向别的内存地址

char\* const pstr2; pstr2是指针常量，const修饰char\*指针，指针不可以指向别的内存地址，但是指向的内容可以使用pstr2来修改

4.volatile关键字的含义

volatile提醒编译器它后面所定义的变量随时都有可能改变，因此编译后的程序每次需要存储或读取这个变量的时候，都会直接从变量地址中读取数据。如果没有volatile关键字，则编译器可能优化读取和存储，可能暂时使用寄存器中的值，如果这个变量由别的程序更新了的话，将出现不一致的现象。

5.static和extern关键字的含义

对于在函数外部定义的static变量，表示这个变量只能在该源代码文件内部使用，对别的源代码文件不可见；如果定义到函数的内部，表示这个变量不是在函数调用的时候创建在栈空间中的变量，而是定义在堆内存中的静态变量，这样每次函数调用的时候，不会重新创建这个变量，而是会保存先前调用这个函数保存的值

|  |
| --- |
| void foo(void){  static int count = 1; //只会第一次调用时候初始化  ...  count ++;  ...  } |

extern关键字，通常用来声明变量，表示这个变量在别的源代码文件中已经定义了，此处给出变量的原型，然后就可以在本文件中使用该变量了（注意此处有个特例：extern int num = 1;如果extern后面有初始化式，那么这个声明同时具有定义的作用）

对于函数定义和声明，类似上面

C语言如果没有上面关键字，默认的链接性是extern的

6.字符串数组和字符串常量

字符串数组：char str1[] = “abcdefg”;

字符串常量：char\* str2 = “abcdefg”;

他们的区别是，str1可以被修改，比如str1[2]=’C’；而如果str2进行这样的错误，就会发生段错误

通常来说，如果定义字符串指针，都是采用const char\* str2 = ”abcdef”;这种形式来定义的

还有注意这个： char\* str\_arr[] = {“aaa”,”bbbbb”}; //sizeof(str\_arr)==8，在str\_arr中实际存放了两个字符串常量指针

7.数组名和指向数组的指针的联系和区别

联系是，对于数组名和指向数组的指针，都代表了数组的首地址；对于函数传参，技术形参是数组名的形式，在函数内部，该数组名也完全退化成指向数组首地址的指针；pstr == arr == &arr[0]

区别是，sizeof数组名返回的是数组实际占用的空间大小；而指针返回的是指针变量占用的大小(32位系统是4字节)；数组名是不可变的，但是指向数组的指针可以改变，比如pstr++指向下一个数组元素

8.函数传参和返回

(1)如果形参是数组名类型int arr[]，arr在函数内部还是会退化成一般类型的指针int\* pint;

(2)传递的参数如果确定在函数内部不会被修改，最好使用const修饰来保护

(3)函数内部定义的数组（以及字符串）都是局部变量，函数调用结束如果返回这个数组的地址是非法的，因为函数调用完，整个栈空间就被清除了；合法的方式是采用malloc从堆中申请内存空间