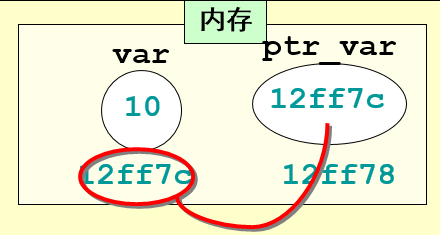
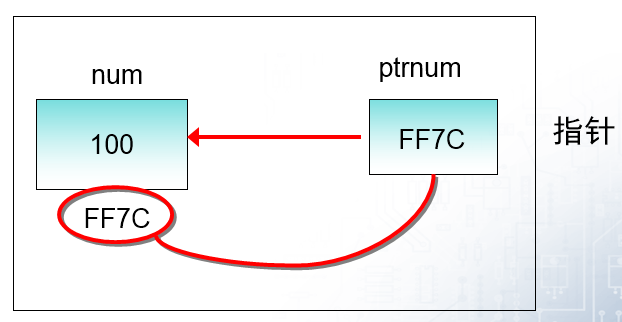
指针也是一个变量（指针变量），只不过该变量中存储的是另一个对象的内存地址

如果一个变量存储另一个对象的地址，则称该变量指向这个对象（即该指针指向某一变量）

由于指针为变量，可以随便赋值和指向。对于指针变量，也存在自己的地址。



值为NULL的指针称为空指针，这意味着，指针并不指向任何地址

若指针已声明为指向某种类型数据的地址，则它不能用于存储其他类型数据的地址

应为指针指定一个地址后，才能在语句中使用指针，避免出现野指针

==================================

int \*ptrnum,arr\_num[8];

ptrnum = &arr\_num[0];

ptrnum++; 🡺 一个类型为 T 的指针的移动，以 sizeof(T)为移动单位

**字符串：**

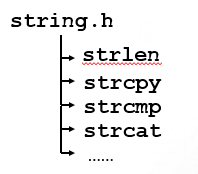
**在Ｃ语言中没有专门的字符串变量，通常用一个字符数组来存放一个字符串**

**字符数组和字符串的区别是：字符串的末尾有一个空字符 ‘\0’**

与字符串有关的内置函数在头文件string.h中定义

要使用标准库字符串处理函数，程序前应该包含：

#include <string.h>



在C语言中，可以将字符指针指向字符串

被指向的字符串可以是一个常量字符串，也可以是一个存储着字符串的字符数组

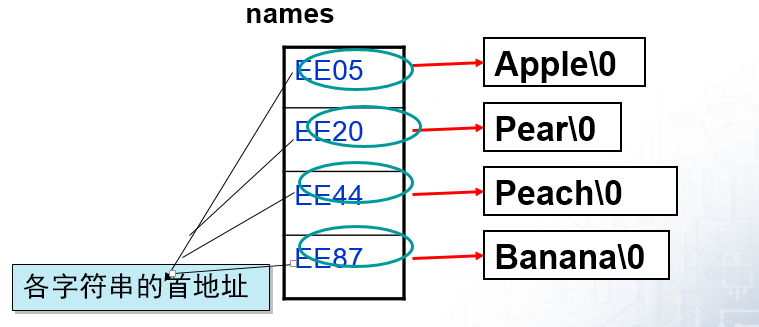
char \*ptr = "C Language";

字符指针数组：一个数组中的各个元素都是字符指针（这些指针变量中存储的为字符串首地址）

char \*names [] = {"Apple", "Pear", "Peach", "Banana"};

以上定义一个字符指针数组，数组中的字符指针指向（存储了）各个字符串首地址。

存储形式：



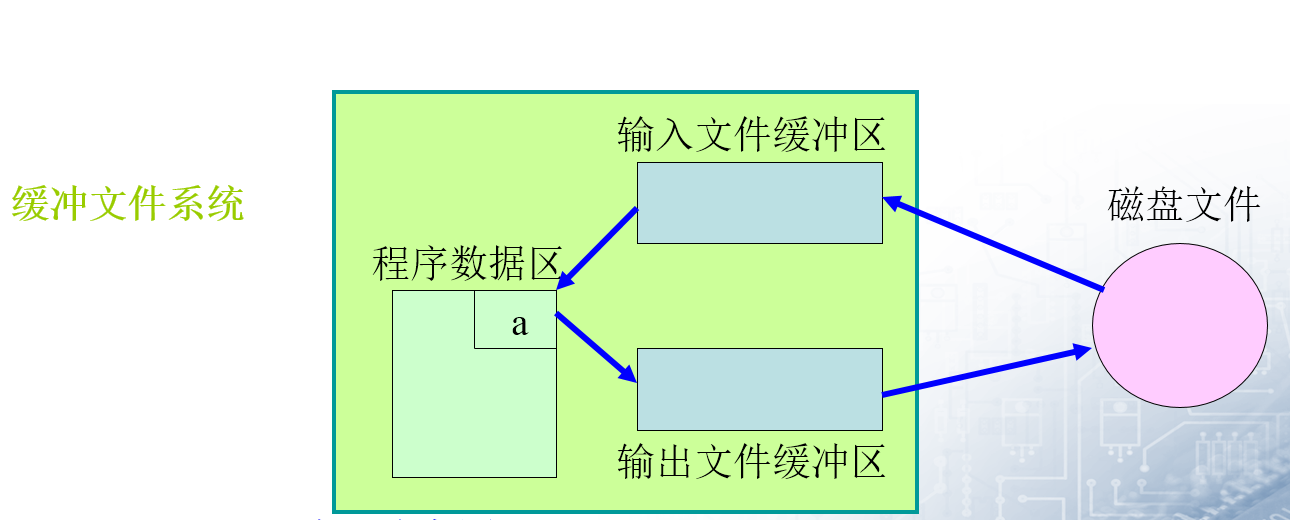
文件：

系统自动打开和关闭的五个**标准设备文件**：



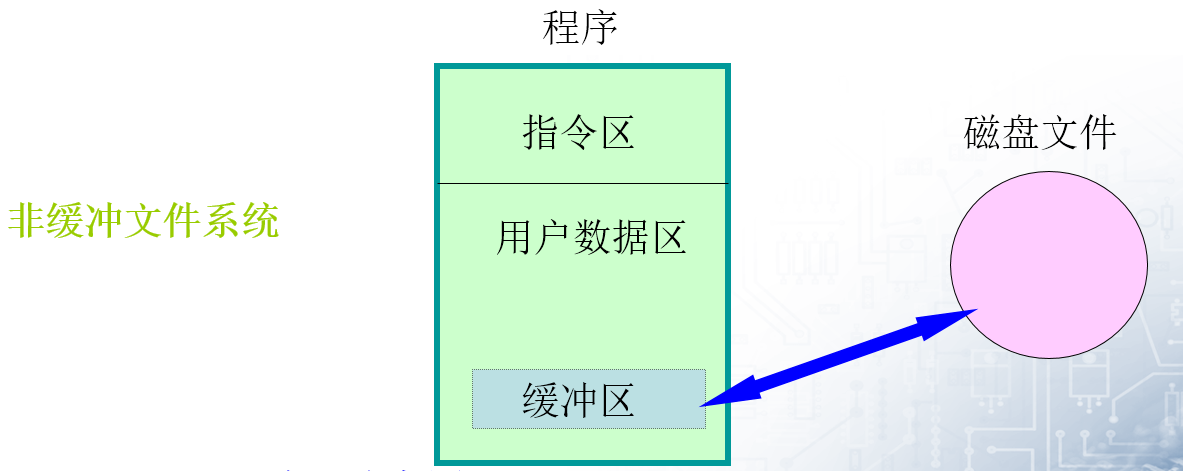
缓冲文件系统（高级文件系统）：

系统自动为正在使用的文件开辟内存缓冲区



非缓冲文件系统（低级文件系统）：

由用户在程序中为每个文件设定缓冲区



如果打开文件进行读操作，该函数函数将清空文件的输入缓冲区；而打开文件进行写操作时，该函数将文件的输出缓冲区写入文件中。

该函数的原型为：

int fflush（FILE \*fp）

清除流

标准输入/输出是经过缓冲的。该函数用来清除缓冲区。如果打开文件进行读操作，该函数函数将清空文件的输入缓冲区；而打开文件进行写操作时，该函数将文件的输出缓冲区写入文件中。

该函数的原型为：

void setbuf(FILE \* stream,char \* buf)；

清空输入缓冲区：

setbuf(stdin, NULL);

清空输出缓冲区：

setbuf(stdout, NULL);

fopen函数：被打开的文件如果在当前目录下，则可以省略路径，否则使用带路径文件名，注意表示各级目录之间使用转义字符‘\\’，而不能使用‘\’。打开文件后，要对返回值进行判断，检查是否打开了文件。只有打开了文件才能继续进行其它操作

对文件进行读写操作时，要注意文件指针的位置，根据要读写的位置，定位文件指针。

C语言高级：

一：

预处理：指是在编译前的处理，包括宏替换，头文件，条件编译#ifdef等。

在标准库中 -- 预定义宏

\_\_DATE\_\_ 进行预处理的日期（“Mmm dd yyyy”形

式的字符串文字）

\_\_FILE\_\_ 代表当前源代码文件名的字符串文字

\_\_LINE\_\_ 代表当前源代码中的行号的整数常量

\_\_TIME\_\_ 源文件编译时间，格式微“hh：mm：ss”

\_\_func\_\_ 当前所在函数名

宏定义中：

操作符#通常称为字符串化的操作符

如：#define mkstr(s) #s

printf(mkstr(I like C));直接输出I like C字符串。

“##”可以把两个独立的字符串连接成一个字符串

#define SORT(X) sort\_function # # X

SORT(3) ---> sort\_function3

二：

异或运算符 -- ^

运算法则：相同为0，不同为1

左移: << 低位补0，高位溢出。

右移：>> 低位放弃，高位正数补0，其他。。。

指针：

int \*p = (int \*)0xF000FF00;

p++(或++p)Æp = p+sizeof(int)，

p--(或--p)->p = p-sizeof(int)。

三：

内存申请：

如果使用动态申请，则申请后一定要判断是否申请成功了，并且

malloc和free应成对出现！

四：c关键字

const --只读

#define

预处理指令

在编译预处理时进行简单的替换，

不作正确性检查

不需要再在内存中分配变量空间

调试程序无法检查用#define说明的常量

用#undef指令取消

typedef

声明一个新的类型名代替已有的类型名

在编译时处理的

不实际分配内存空间

#define your\_int int \*

your\_int a, b;

即

int \*a;

int b;

typedef int \* your\_int;

your\_int a, b

即

int \*a;

int \*b;

static 的作用

在函数体，一个被声明为静态的变量在这一函数被调用过程中维持

其值不变。

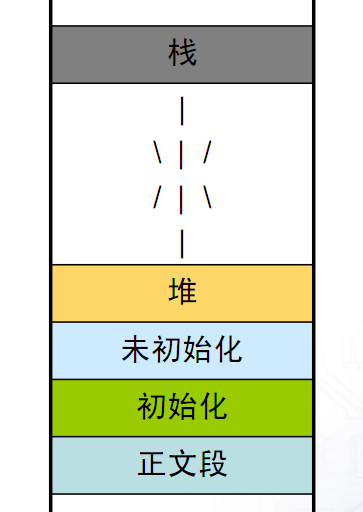
在模块内（但在函数体外），一个被声明为静态的变量可以被模块

内所用函数访问，但不能被模块外其它函数访问。

在模块内，一个被声明为静态的函数只可被这一模块内的其它函数

调用

C语言的存储布局（c应用的存储布局）



Heap与stack

Stack的空间由操作系统自动分配/释放，Heap上的空间手动分配/释

放。

Stack空间有限，Heap是很大的自由存储区

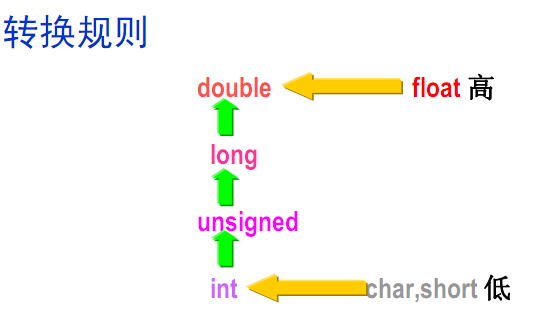
内存分配域

malloc函数分配的内存空间即在堆上

程序在编译期对变量和函数分配内存都在栈上进行

程序运行过程中函数调用时参数的传递也在栈上进行

强制类型数据转换：



横向箭头表示必须的转换

纵向箭头表示当运算符两边的运算数为不同类型时的转换

低－》高 ＝＝》形式上改变，数据的实质内容不变

高－》低 ＝＝》可能有些数据丢失

C中的错误点：

“=”是赋值运算符

“==”是关系运算符