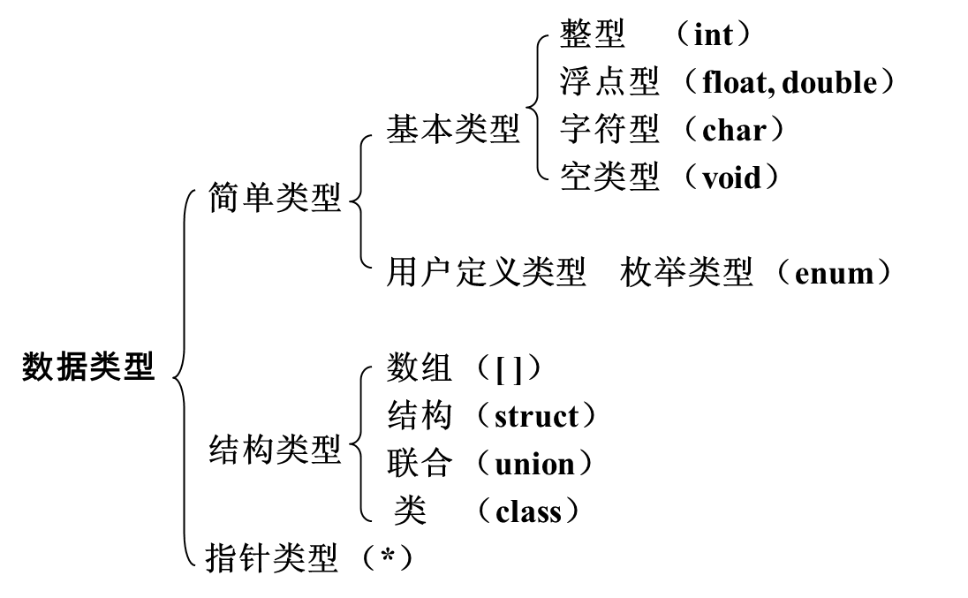
1. **基础知识**
2. 数据类型：对数据的抽象



1. int b[10]

b:代表数组首元素的地址

&b:代表整个数组的地址

1. 变量的本质：

1、程序通过变量来申请和命名内存空间，eg：int a = 0

**2、通过变量名访问内存空间**

（一段连续）内存空间的别名（是一个门牌号）

**修改变量有几种方法？**

1. **直接：**

p = 200;

1. **间接**。内存有地址编号，拿到地址编号也可以修改内存

**(1)通过内存赋值**

\*((int\*)1245024) = 200;

**(2)通过指针赋值**

p = 1245024;

\*p = 200;

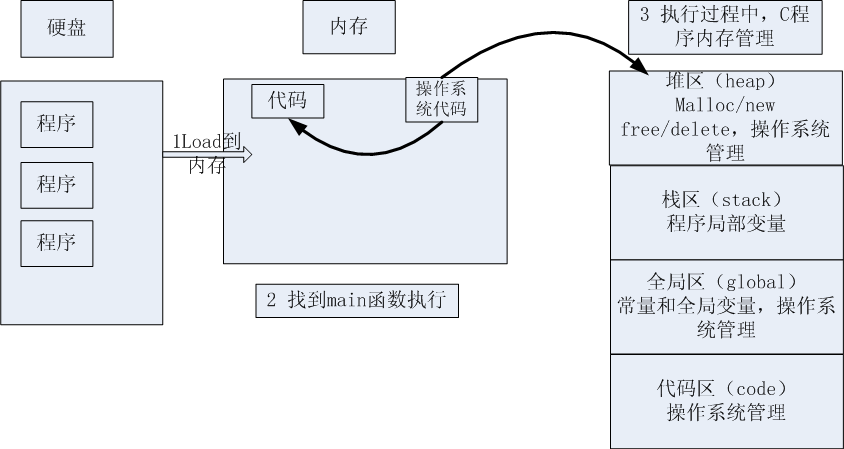
3、内存空间可以再取给别名吗？

1. 运行内存布局图：

内存四区模型

函数调用模型

1. 内存四区模型



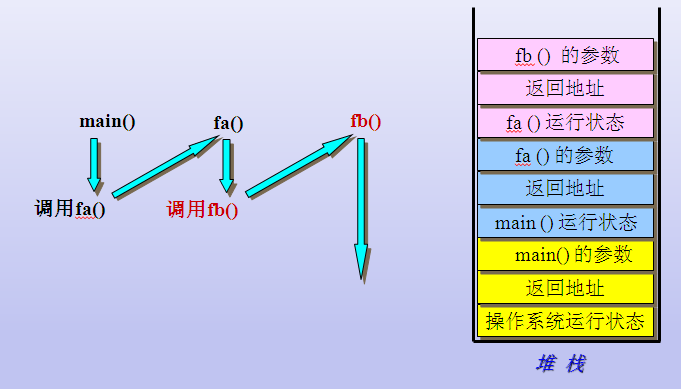
**流程说明**

1、操作系统把物理硬盘代码load到内存

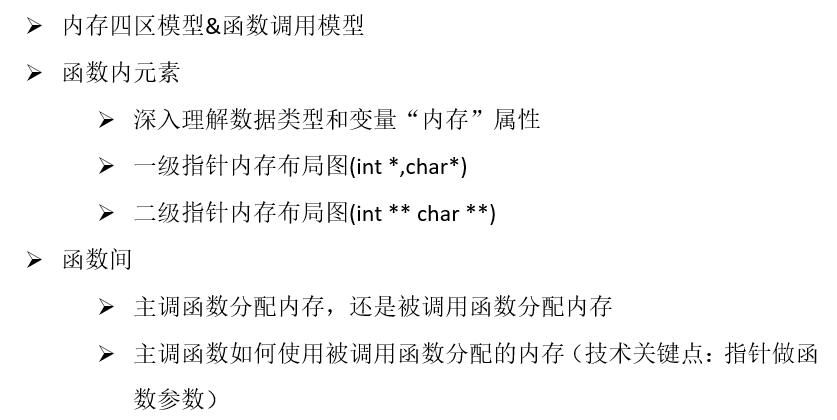
2、操作系统把c代码分成四个区

3、操作系统找到main函数入口执行

1. 函数调用模型



1. main函数可以在堆、栈、全局区分配内存
2. 所以，main的内存可以被fa()、fb()使用。
3. 建立内存布局图



1. **指针**
2. 指针也是一种数据类型
3. 不断的给指针赋值，等同于不断的改变指针的指向
4. **\*p操作内存**

在指针声明时，\*号表示所声明的变量为指针

在指针使用时，\*号表示操作指针所指向的内存空间中的值

\*p相当于通过地址(p变量的值)找到一块内存；然后操作内存

\*p放在等号的**左边赋值**（给内存赋值）

\*p放在等号的**右边取值**（从内存获取值）

1. 指针变量和它指向的内存块是两个不同的概念

//含义1 给p赋值p=0x1111; 只会改变指针变量值，不会改变所指的内容；p = p +1; //p++

//含义2 给\*p赋值\*p='a'; 不会改变指针变量的值，只会改变所指的内存块的值

//含义3 =左边\*p 表示 给内存赋值， =右边\*p 表示取值 含义不同切结！

//含义4 =左边char \*p

//含义5 保证所指的内存块能修改

1. **指针是一种数据类型**，是**指它指向的内存空间的数据类型**

含义1：指针步长（p++），根据所致内存空间的数据类型来确定

p++=🡺(unsigned char )p+sizeof(a);

结论：指针的步长，根据所指内存空间类型来定。

1. 指针数组：

int \*a[10];

一个有10个指针的数组，该指针是指向一个整型数的。

1. 数组指针：

int (\*a)[10];

一个指向有10个整型数数组的指针

1. 函数指针：

int (\*a)(int);

一个指向函数的指针，该函数有一个整型参数并返回一个整型数。

1. **关键字static的作用是什么？**
2. 在函数内，静态变量在该函数被调用过程中，能够维持数据不变
3. 在模块内（但不在函数内），静态变量可被模块内的函数调用，是一个本地的全局变量。
4. 在模块内，静态函数只能在模块内使用，限定使用范围。

**总结：**

1. static用来修饰一个局部的变量的时候，  
    生命域是全局的  
    作用域是局部的
2. static用来修饰一个模块内的（某一个C的源程序文件）全局变量的时候  
     生命域不变  
     作用域减小，只在本模块内有效
3. static用来修饰一个函数的时候  
     作用域减小，只在本模块内有效

**关键的区别：**

（1）static全局变量与普通的全局变量有什么区别：  
 static全局变量只初使化一次，防止在其他文件单元中被引用

（2）static局部变量和普通局部变量有什么区别：  
 static局部变量只被初始化一次，下一次依据上一次结果值；

（3） static函数与普通函数有什么区别：  
 static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

1. 关键字const有什么含意？

**constant**：只读。**“常“：**意味着，不能修改。

**作用：**

1. 使用关键字const也许能产生更紧凑的代码。
2. 关键字const使编译器保护那些不希望被改变的参数，减少bug的出现。

**eg：**

const int a; a是一个常整型数  
int const a; a是一个常整型数  
const int \*a; a是一个指向**常整型数**的**指针**，整型数是不可修改的，但指针可以  
int \* const a; a：指向**整型数**的**常指针**，针指向的整型数可以修改，但指针不可修改的  
int const \* a const; a是一个指向**常整型数**的**常指针**，指针指向的整型数是不可修改的，

同时指针也是不可修改的。

1. 关键字volatile

volatile的变量，表示被修饰的符号是易变的。优化器使用volatile变量时，都需要重新读取该变量的值，不使用寄存器内的备份。

**应用场合：**  
 \*一个硬件寄存器  
 \*中断中用到的变量  
 \*线程之间共享变量

**Q：**

1. 一个参数既可以是const还可以是volatile吗？

是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。

2. 位操作：

Q：给定一个整型变量a，写两段代码，第一个设置a的bit 3，第二个清除a 的bit 3。

A：

#define BIT3(0x1 << 3)

static int a;

void set\_bit3(void)

{

a |= BIT3;

}

void clear\_bit3(void)

{

a &= ~BIT3;}

1. malloc（0）、NULL区别

malloc(0)是指分配内存大小为零   
NULL是不指向任何实体   
malloc(0)也是一种存在不是NULL

1. **C和C++的不同**  
    1）c++源于c，c++最重要的特性就是引入了面向对象机制，class关键字。  
    2）c++中，变量可以再任何地方声明；c中，局部变量只能在函数开头声明。  
    3）c++中，const型常量是编译时常量；c中，const常量只是只读的变量。  
    4）c++有&引用;c没有  
    5）c++的struct声明自动将结构类型名typedef；c中struct的名字只在结构标签名字

空间中，不是作为一种类型出现  
 6）c语言的main函数可以递归调用;c++中则不可以  
 7）c中，void \*可以隐式转换成其他指针类型；c++中要求现实转换，否则编译通不过

1. 枚举：

枚举类型(enumeration)是C++中的一种派生数据类型，它是由用户定义的若干枚举常量的集合。如果一个变量只有几种可能的值，可以定义为枚举(enumeration)类型。所谓"枚举"是指将变量的值一一列举出来**，变量的值只能在列举出来的值的范围内**。

**模板：**

enum 枚举名{

标识符[=整型常数],

标识符[=整型常数],

...

标识符[=整型常数]

} 枚举变量;

eg:

enum color { red, green, blue } c;

c = blue;

2. **例题：**
3. **有符号、无符号参数的运算**

void foo(void)  
{  
    unsigned int a = 6;  
    int b = -20;  
    (a+b > 6) ? puts("> 6") : puts("<= 6");  
}

输出：>6

**原因：**C语言中的整数自动转换原则，**当表达式中存在有符号类型和无符号类型时所有的操作数都自动转换为无符号类型**。因此-20变成了一个非常大的正整数。

1. 用宏定义求解数组Table的元素个数：

COUNT(Table) (sizeof(Table) / sizeof(Table[0]));

1. 用宏定义求解a，b中的最大值：

#define MAX(a,b) ((a > b) ? (a) : (b));

1. 定义结构体：

//定义

typedef struct Teacher

{

char name[64];

int age;

}Teacher;

//使用

Teacher teacher;