内存映像

程序：是存储在磁盘上的可执行文件(二进制文件、脚本文件)

进程：正在系统中运行的程序，可以存在多个进程

进程映像就是进程在系统的中内存分布情况：

text 代码段 存储的是二进制指令、常量，权限是只读，如果强制修改会产生段错误；

data 数据段 存储初始化过的全局变量、初始化过的静态局部变量；

bss 静态数据段 存储未初始化过的全局变量、未初始化过的静态局部变量，在程序运行前该段内存会自动清理为0；

stack 栈 存储局部变量、块变量，会随着进程的运行而不断申请、释放，是由操作系统负责管理，使用方便，缺点：小；

heap 堆 可以存储大量的数据，特点足够大，由程序员手动管理，使用不方便；

局部变量和全局变量：

局部变量：定义在函数内

存储位置：stack 栈内存

生命周期：从函数调用开始到函数结束

作用范围：只能在函数内使用

全局变量：定义在函数外

存储位置：data(初始化)或者bss(未初始化)

生命周期：从main运行前到程序结束后才释放

作用范围：程序的任何位置都可以使用

块变量:定义在if\for\while语句块内

存储位置：stack 栈内存

生命周期：语句块调用开始到语句块结束

作用范围：只能在语句块内使用

注意：局部变量和全局变量可以同名，但是同名局部变量会屏蔽同名的局部变量，快变量也会屏蔽同名的局部变量和全局变量

注意：建议全局变量首字母大写

类型限定符：

auto

用于定义自动申请、自动释放内存的变量(局部变量)，不加就代表了加，

不用使用auto修饰全局变量

C11中的auto用于自动定义类型

const

用于"保护"变量不被显式的修改

注意：如果用const修饰初始化过的全局变量(data)，存储位置会改为代码段(text)，此时就不能强制修改了

extern

用于声明变量

被extern声明过的变量，表示该变量已经在别的地方定义过了，请放心使用

但是只能临时瞒过编译器，让代码通过编译，但是通过链接是找不到该变量依然会报错

不能对extern修饰的变量赋值

static

存储位置：

改变局部变量的存储位置，由stack改为data(初始化)或者bss(未初始化)

被它修饰过的变量也叫静态变量

生命周期：

延长局部变量的生命周期

作用范围：

限制全局变量、函数的作用范围，只能在本文件内使用

可以防止全局变量、函数名命名冲突，防止被别人使用

volatile

如果变量的值没有显式的发生改变，那么在使用该变量时，编译器不会从内存中读取该变量，而是继续使用第一次读取到的结果，以此提高运行速度，这叫取值优化

如果使用volatile修饰变量，那么每次使用该变量时都会从内存中读取，不做取值优化

一般在硬件编程中使用

————————————————

版权声明：本文为CSDN博主「hklsjt」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/hklsjt/article/details/121255831