**C 程序是如何在硬件上运行的**

C 语言的编译链接过程是要把编写的源代码转换成可在硬件上运行的可执行代码。

编译就是将文本形式的源码翻译为机器语言形式的目标文件的过程。

链接就是将目标文件、启动代码以及其他用到的库文件进行组织，最终形成可执行代码的过程。

预编译.i (gcc -E)

编译 .s (gcc -s)

编译器

编译过程

gcc -c 汇编命令 as

汇编器

其他lib

链接器

链接命令ld

可执行程序

预处理阶段

宏定义、条件编译指令、头文件、特殊字符等的展开

编译优化阶段

语法分析检查等

汇编

形成目标机器指令的文件，由段组成。至少包含代码段、数据段。

代码段：主要是程序的指令，一般只读

数据段：程序用到的各种变量与数据

链接

有关的目标文件进行链接，分静态链接、动态链接。

静态：拷贝到可执行程序中

动态：登记信息，运行时寻找

**ld 文件的作用**

ld 文件是一种链接器使用的规则性文件，告诉链接器系统的 ROM RAM 的地址信息以及大小，并指示将什么代码放置在什么地方。

**内存四区**

C 语言中的内存四区：堆区、栈区、全局区、代码区

堆区：由程序编写进行申请释放

栈区：由编译器自动分配释放

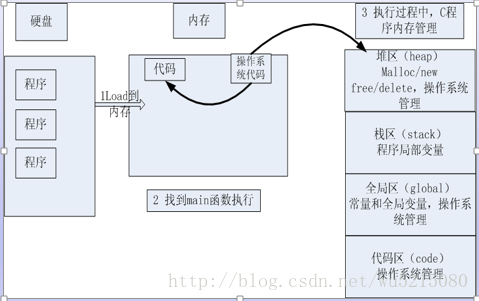
全局区：全局变量和静态变量的存储放在一起，bss

初始化的全局变量和未初始化的静态变量在相邻的另一块区域，

字符串常量和其他常量存储在常量区，

操作系统管理

代码区：存放程序的二进制代码，由操作系统管理



操作系统把硬盘中的数据下载到内存，并将内存划分成四个区域，由操作系统找到main入口开始执行程序。

一、栈区

由编译器自动分配释放，存放函数的形参、局部变量等。当函数执行完毕时自动释放。

#include<stdio.h>

char \*getMem()

{

char buf[64]; //局部变量 栈区存放

strcpy(buf, "123456789");//向buf所代表的内存中写入内容

//printf("buf:%s\n", buf);

return buf;//返回所分配内存区域的第一个元素的地址

}

void main()

{

char \*tmp = NULL;

tmp = getMem(10);

if (tmp == NULL)

{

return ;

}

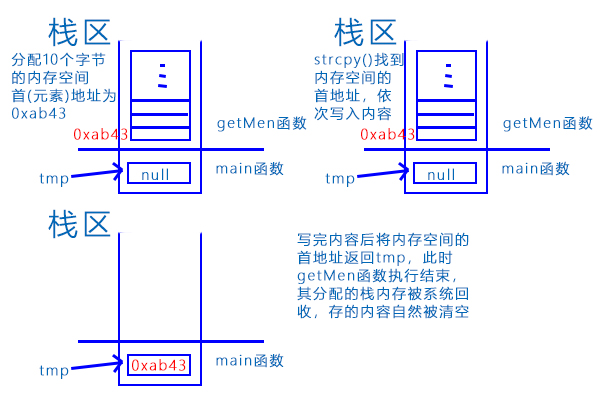
printf("tmp:%s\n", tmp);//输出tmp:

system("pause");

return ;

}

内存分析：



二、堆区

一般由程序员手动分配释放（动态内存申请与释放），若程序员不释放，程序结束时可能由操作系统回收。

#include <stdio.h>

char \*getMem(int num)

{

char \*p1 = NULL;

p1 = (char \*)malloc(sizeof(char) \* num);

if (p1 == NULL)

{

return NULL;

}

return p1;

}

void main()

{

char \*tmp = NULL;

tmp = getMem(10);

if (tmp == NULL)

{

return ;

}

strcpy(tmp, "111222"); //向tmp做指向的内存空间中copy数据，注意不是向指针变量tmp中

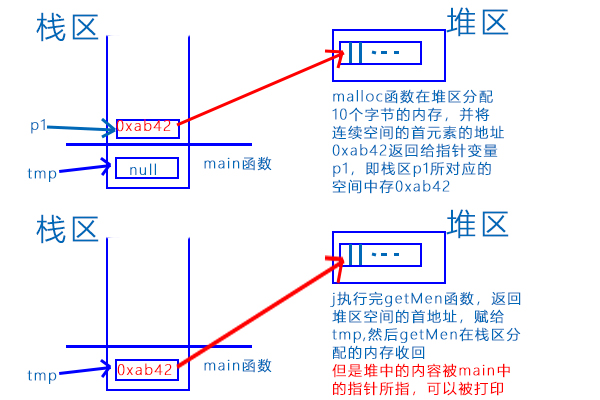
printf("tmp:%s\n", tmp);//输出tmp:111222

system("pause");

return ;

}

内存分析：



三、全局区

用于存放全局变量和静态变量， 里面细分有一个常量区，字符串常量和其他常量也存放在此。该区域是在程序结束后由操作系统释放。

#include <stdio.h>

char \* getStr1()

{

char \*p1 = "abcd";

return p1;

}

char \*getStr2()

{

char \*p2 = "abcd";

return p2;

}

void main()

{

char \*p1 = NULL;

char \*p2 = NULL;

p1 = getStr1();

p2 = getStr2();

//打印p1 p2 所指向内存空间的数据，不是p1 p2中的数据

printf("p1:%s , p2:%s \n", p1, p2);//输出p1:abcd , p2:abcd

//打印p1 p2 的值

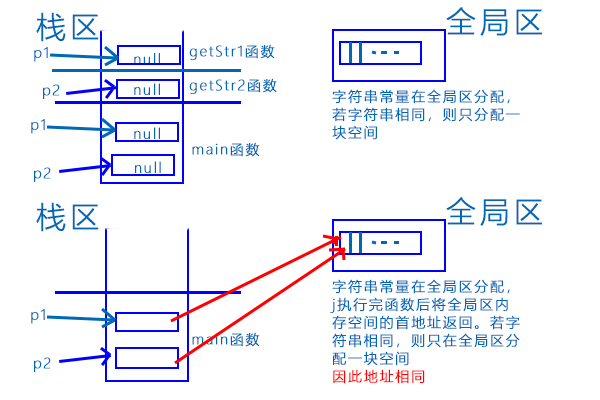
printf("p1:%d , p2:%d \n", p1, p2);//输出p1:19184372 , p2:19184372

system("pause");

return;

}

内存分析：



\*函数调用：

1.在程序未执行结束时，main函数里分配的空间均可以被其他自定义函数访问；

2.自定义函数若在堆区（malloc动态分配内存等）或全局区（字符串常量等）分配的内存，即便此函数结束，这些内存空间也不会被系统回收，内存中的内容可以被其他自定义函数和main函数使用