琴鸟

c 函数调用产生的汇编指令和数据在内存情况(2)

c 函数调用产生的汇编指令和数据在内存情况(1)

一直对函数调用的具体汇编指令和各种变量在内存的具体分配，一知半解。各种资料都很详细，但是不实践，不亲自查看下内存总不能笃定。那就自己做下。

两个目的：

一，函数和函数调用编译后的汇编指令基本样貌

二，各种变量类型的内存状况。

二，各种变量类型的内存状况。

1)常见变量在内存的位置

昵称：琴鸟
园龄：7年8个月
粉丝：2
关注：1
+加关注

2)自定义结构体

1)，常见变量在内存的位置。

结论：全局变量：程序一加载，和代码一样，已经在内存，放入静态区。

未初始化，内存数据用00或默认直代替。

地址变量(指针类型)放入地址直。

未初始化放入0x00000000.

局部变量： int 和char 等基本类型，程序加载时，不放入任何地方。

只有通过代码才能知道定义了一个变量。

运行代码时 push 1，放入栈中，通过 ebp+x等方式获取。

而int[5] 和char[5] 类拭固定大小数据，一般是放入静态区，编译器在编译阶段已经把使用变量的地方用 变量的偏移地址代替了。如果函数没使用，直接作为其他函数的参数，那也会直接push。不放入静态区，如下例的 p_char2[5]。

不是固定大小的变量放入，如指针，放入静态区。。等等，如果中途改变大小呢，怎么办？等下测试。测试发现会有2个临时变量名。

```
char * p_char3="hi.";
int p_int2[5]={1,2,3,4,5};
p_char3="hihi.";
```

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

我的标签

c(1)
p民(1)
编译(1)
汇编(1)
内存(1)

随笔分类

.net(14)
c++(5)
好文转载
计算机系统
解惑(17)
烂尾的东西(2)
数据库(4)
算法(2)

随笔档案

2016年11月 (10)
2016年10月 (12)
2016年9月 (2)
2016年8月 (1)
2016年7月 (2)
2016年6月 (2)
2016年5月 (6)

```
LC2:
        DB      "hihi.",0x00
LC0:
        DB      "hi.",0x00
```

代码

```
int g_int1=3;//静态区.装载程序时已经放入内存
int g_int2;//静态区.装载程序时已经放入内存(放在 char * p_char3="hi."的后面).用4个字节的0来占位。

int HariMain(void)
{
    int p_int=1;//代码没有执行，不存在任何地方，执行后，push 1,放入栈中。
    char p_char='a';// 代码没有执行，不存在任何地方，执行后，push 1,放入栈中。
    char p_char2[5]={'a','b','c','d','e'};//
    //代码没有执行，不存在任何地方，执行后，用mov指令放入栈.
    //mov BYTE [-56+EBP],97
   char
* p_char3="hi.";//静态区. 装载程序时已经放入内存
   int p_int2[5]={1,2,3,4,5};//静态区. 装载程序时已经放入内存
```


文章分类

最新评论

```
    unsigned int sum;
    sum=count(p_int,p_char,p_char2,p_char3,p_int2);
    sum+=g_int1;
    sum+=g_int2;
}

unsigned int count(int a,char c1,char c2[5],char * c3,int i2[5])
{
    unsigned int c;
    c=0;
    c+=a;
    c=c+c1;
    c+=c2[0];
    c+=c2[3];
    c+=i2[0];
    c+=i2[4];
    c+=c3[0];
    c+=c3[1];
    return c;
}
```

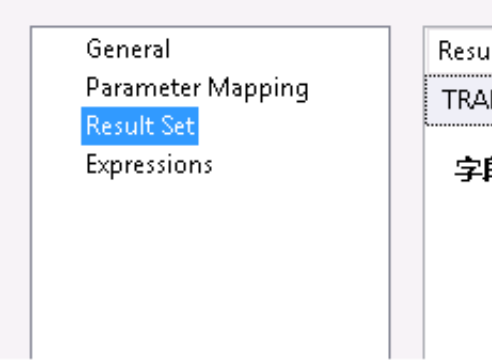
数据 在内存的位置

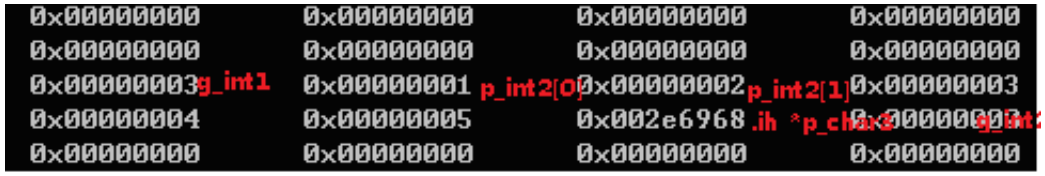
的存在形式。

基本数据类型.int ,char short.int a;a 标签代表一个地址的数据,里面的数据类型是int.所以占4个字节.a=3;给基本数据类型的标签赋直.就等于给标签代表的地址的数据赋直.a 标.....

--琴鸟

2. Re:sql for xml 嵌套





程序装载时,

//代码区 0x0028001b

//

//^

//栈顶(空栈) 0x00310000

//静态区　　0x00310000

程序运行时

//代码区 0x0028001b

/

//(栈顶)被调者的临时变量

//被调者的局部变量

//调用前的ebp寄存器直(而当前的ebp寄存器存放的这个位置的地址)

//返回地址

//参数

//栈底(空栈) 0x00310000

--琴鸟

3. Re:转 快速建立Subversion

svn,地址

svn://ip/svn

--琴鸟

4. Re:转 快速建立Subversion

和vs 配合 如何 给所有文件加上 lock 属性。先直接加入 项目到 svn。这个时候没有加锁转到 项目根目录。右键 , svn 菜单。属性。新加入 needs-lock 属性。关闭 vs。开启 vs.....

--琴鸟

5. Re:转 快速建立Subversion

1. 安装 SubversionC:\Program Files\Subversion2.建立目录存放文档数据E:\project\svnproject是我们所有项目的文档目录。svn是我们第一个项.....

--琴鸟

阅读排行榜

//静态区　0x00310000

二，自定义结构体

结论：自定义结构，可以看作数组。

自定义结构，作为参数的话，会把所有成员变量，一个一个入栈

如果 传递自定义结构指针，那么只传地址。

//全局自定义结构体变量，和全局定长数组类拭。

程序一加载，和代码一样，已经在内存，放入静态区。

未初始化放入00数据，

代码中出现变量名，用地址代替。[_struce_a]

赋直：

```
MOV     BYTE [_myStruck_a+4],97
```

直接　地址+数字定位成员

全局自定义结构体地址变量(指针)，

程序一加载，和代码一样，已经在内存，放入静态区。

但是大小不是struck的大小，而是4B，也就是一个地址变量的大小。

未初始化放入0x00000000.

赋直：

```
MOV     EDX,DWORD [_myStruck_c]
MOV     DWORD [8+EDX],3
```

必须取地址的直得到真正的地址再加数字定位成员

//局部变量,

程序一加载，不存在任何地方。

只有运行时，放入栈中。如：

```
struct myStruck myStruck_d;
```

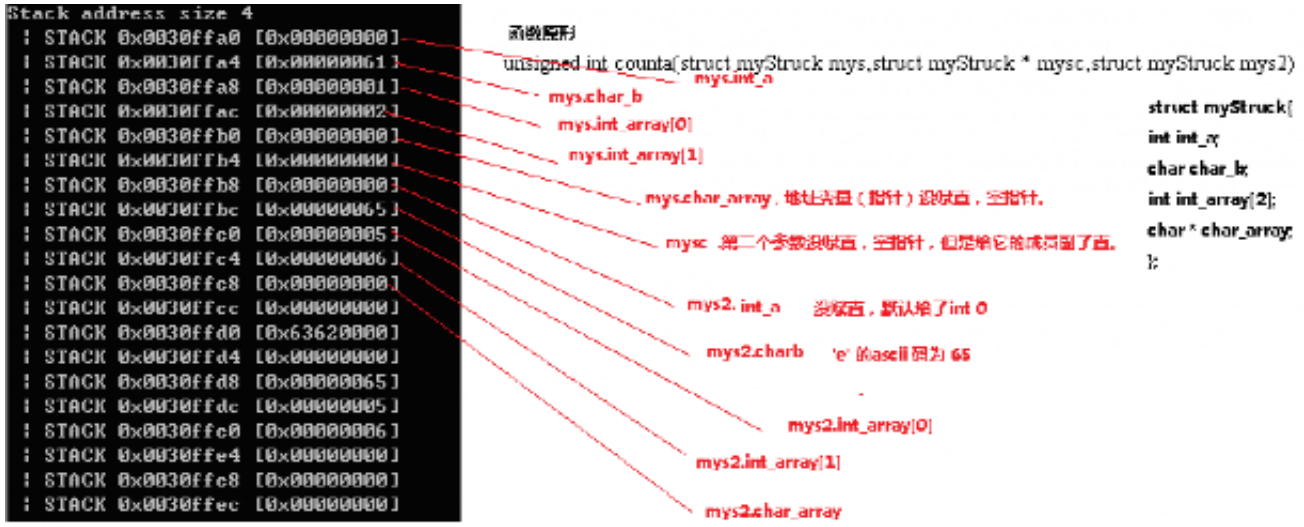
编译为SUB ESP,60

```
myStruck_d.char_b='e'
```

编译为

```
MOV  BYTE [-36+EBP],101
```

call 之后的栈数据.




```
struct myStruck{
int int_a;
char char_b;
int int_array[2];
char * char_array;
};
int HariMain(void)
{
    char c1[2]={'b','c'};
```

```
    //myStruck_a.int_a=1;
    myStruck_a.char_b='a';//MOV       BYTE [_myStruck_a+4],97
    myStruck_a.int_array[0]=1;//MOV       DWORD [_myStruck_a+8],1
    myStruck_a.int_array[1]=2;//MOV       DWORD [_myStruck_a+12],2
    //myStruck_a.char_array=c1;



//MOV       EAX,DWORD [_myStruck_c]
    myStruck_c->int_a=2;//MOV       DWORD [EAX],2
    myStruck_c->char_b='c';//MOV       BYTE [4+EAX],99

//MOV       EDX,DWORD [_myStruck_c]
    myStruck_c->int_array[0]=3;//MOV       DWORD [8+EDX],3
    myStruck_c->int_array[1]=4;//MOV       DWORD [12+EDX],4
    myStruck_c->char_array=c1;//LEA       EAX,DWORD [-42+EBP]     MOV       DWORD
[16+EDX],EAX



//MOV       EBP,ESP
//SUB       ESP,60
    struct myStruck myStruck_d;

    myStruck_d.char_b='e';//MOV       BYTE [-36+EBP],101
```

```
    myStruck_d.int_array[0]=5;//MOV       DWORD [-32+EBP],5
    myStruck_d.int_array[1]=6;//MOV       DWORD [-28+EBP],6



    unsigned int c=counta(myStruck_a,myStruck_c,myStruck_d);


    io_hlt();
    return 0;
}


unsigned int counta(struct myStruck mys,struct myStruck * mysc,struct myStruck mys2)
{
    unsigned int c;
    c=0;
    c=mys.int_a;
    c=c+mysc->int_a;//ADD       EAX,DWORD [8+EBP]
    c=c+mysc->char_array[0];//MOV       EDX,DWORD [28+EBP]  MOV       EDX,DWORD
[16+EDX]  ADD       EAX,EDX  //char_array[0]
    c=c+mys.char_array[0];//
    c=c+mys2.int_a;
    return c;
}
```

分类: 解惑

琴鸟
关注 - 1
粉丝 - 2
+加关注

0 0

« 上一篇：c 函数调用产生的汇编指令和数据在内存情况(1)
» 下一篇：hanio 塔和递规的理解。

posted @ 2016-05-08 18:38 琴鸟 阅读(30) 评论(0) 编辑 收藏