# C++ Tricks 2.7 I386 平台的其它函数调 用模型 🗵

从 farseerfc.wordpress.com 导入

# 2.7 I386平台的其它函数调 用模型

上文介绍的只是I386平台上C函数调用的标准模型,被称作\_\_cdecl。事实上,Microsoft Visual C++编译器还支持其它一些函数调用模型,所有调用模型名称皆以双下划线开头,下面列出所有函数调用模型的异同:

#### 1 \_\_cdecl

参数压栈顺序:逆序(从右至左)

参数堆栈恢复者:主调函数(caller)

\_\_cdecl明确地指出函数使用C函数调用模型,这是默认的调用模型。

#### 2 stdcall

参数压栈顺序:逆序(从右至左)

参数堆栈恢复者:被调函数(callee)

\_\_stdcall是微软所谓的标准调用模型。可惜的是它与 \_\_cdecl不兼容。几乎所有的Win32API函数使用这种函 数调用模型,希望在DLL之间,或者在程序和WinNT操作 系统之间传递函数指针的函数也应该使用这种模型。与 \_\_cdecl模型的不同之处在于,\_\_stdcall模型下由被调函 数恢复堆栈。主调函数在call语句之后,不需要再加上 add语句。而被调函数的ret语句则被添加一个参数,代表函数参数堆栈的长度。因此,被调函数需要明确的知晓函数参数的数量和类型,所以在\_\_stdcall模型下不支持可变参数表,所有参数必须写明。

#### 3 \_\_thiscall

参数压栈顺序:逆序(从右至左),this用ecx传递。

参数堆栈恢复者:被调函数(callee)

\_\_thiscall是VC编译器中类的非静态成员函数(non-static member functon)的默认调用模型。但是如果此成员函数有可变参数表,VC编译器会使用\_\_cdecl。和\_\_stdcall一样,\_\_thiscall由被调函数恢复堆栈。比较独特的是\_\_thiscall会通过ecx寄存器传递成员函数的this指针,而\_\_cdecl下this指针是通过在参数表最前面增加一个函数参数来传递的。\_\_thiscall是VC编译器对this指针的使用的一种优化,大大提高了面向对象程序的效率。在VC2003及之前的编译器上\_\_thiscall不是一个关键字,不能被显式指定。但可以给成员函数显式指定\_\_cdecl来避免使用\_\_thiscall。

## 4 \_\_fastcall

参数压栈顺序:逆序(从右至左),前两个32位函数参数放入ecx和edx中

参数堆栈恢复者:被调函数(callee)

快速函数调用模型,将前两个32位函数参数放入ecx和edx中,其余参数再逆序压栈。使用的是和\_\_thiscall类似的优化技术,加快函数调用,适合运用在小型inline函数上。同样使用\_\_stdcall形式的被调函数恢复堆栈,所以不支持可变参数表。

## 5 \_\_pascal

参数压栈顺序:正序(从左至右)

参数堆栈恢复者:被调函数(callee)

过程式编程语言Pascal所使用的函数调用模型,由此得名。也是16位版本的Windows使用的API模型,过时的模型,现在已经废弃且禁止使用。你会看到有些书本仍会不时提到它,所以需要注意。\_\_pascal是正序压栈,这与大部分I386函数模型都不相同。与\_\_stdcall一样,由被调者恢复堆栈,不支持可变参数表。历史上曾有过的别名PASCAL、pascal、\_pascal(单下划线),现在都改成了\_\_stdcall的别名,与\_\_pascal(双下划线)不同。

#### 6 其它函数调用模型,以及模型 别名。

\_\_syscall:操作系统内部使用的函数调用模型,由用户模式向核心模式跳转时使用的模型。由于用户模式和核心模式使用不同的栈,所以没办法使用栈来传递参数,所有参数通过寄存器传递,这限制了参数的数量。用户模式编程中不允许使用。

\_\_\_fortran:数学运算语言fortran使用的函数模型, 由此得名。在C中调用由fortran编译的函数时使用。

\_\_clrcall:微软.Net框架使用的函数模型,托管 (Managed)C++默认使用,也可以从非托管代码调用托管 函数时使用。参数在托管栈上正序(从左至右)压栈,不使用普通栈。

CALLBACK、PASCAL、WINAPI、APIENTRY、APIPRIVATE:I386平台上是\_\_stdcall的别名

WINAPIV:I386平台上是\_\_cdecl的别名

#### 7 函数调用模型的指定

函数调用模型的指定方式和inline关键字的指定方式相同,事实上,inline可以被看作是C++语言内建的一种函数调用模型。唯一不同的是,声明函数指针时,也要

指明函数调用模型,而inline的指针是不能指明的,根本不存在指向inline函数的指针。比如:

int CALLBACK GetVersion();

int (CALLBACK \* pf)()=GetVersion;