

## C++ Tricks 2.7 I386 平台的其它函数调用模型

从 [farseerfc.wordpress.com](http://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 function) 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;
```