Windows 异常处理机制介绍

http://hi.baidu.com/xplot/blog/item/9e7f3154b6e635c2b645ae32.html

2008-10-14 17:06

最近做了一个 Windows 下的异常处理模块，查阅了一些新的资料，结合我自己的理解，将一些点滴记录如下，希望对兄弟们有所帮助。

一、C++标准异常

也就是 try、throw、catch 这三个关键字。

```
try
{
    ……
    throw <exception-data>
    ……
}

catch (<exception-declaration 1>)
{
    ……
}
catch (<exception-declaration 2>)
{
    ……
}
……
```

try 块中的 throw 会抛出一个数据<exception-data>，比如一个整数，一个字串，或是其他自定义类型的数据。这时，当前程序中止执行，开始查找 catch 入口。throw 抛出的数据类型与 catch 入口的<exception-declaration>数据类型必须匹配，这一点类似函数调用的形参、实参匹配。一个 try 块可以对应多个 catch 块，这一点类似于函数的重载。当然，你也可以用 catch (…)来接收所有可能抛出的数据。MFC 提供了一些标准的抛出异常类型，如 CFileException 类、CDaoException 类等，它们都是 CException 类的派生类，使用 MFC 时可以了解一下，这里就不多说。

执行完 catch 块，程序会继续向下执行。

当 throw 在本函数（或说栈的本帧）没有找到合适的 catch 块时，会向上一层调用函数（或说栈的上一帧）回溯，直到匹配到合适的 try-catch 块为止。也就是说，try-catch 块可以捕获到 try 块中调用(可以是多层调用)的函数中的，没被处理的异常。同时，try-catch 块是可以嵌套的。

那么，有一个问题：没有 try-catch 块的，或查找到调用顶层（如 main 函数）都没有匹配上 catch 块的 throw 语句会如何执行呢？这在不同操作系统会有不同的处理，在 Windows 中则是由一个叫 SEH 的机制来处理的。

二、Windows SEH

SEH（Structured Exception Handling），即结构化异常处理，是 Microsoft 提供的异常处理机制。要了解这个机制，咱先来了解一下__try-__except 关键字。

1. __try-__except 关键字

```
__try
{
    ……
}

__except (<exception>)
{
    ……
}
```

__try-__except 是 Microsoft 扩展出的 C++关键字，__try 块中出现错误或异常，一般不再用 throw 抛出，而是直接产生一个 EXCEPTION_POINTERS 类型的异常数据，然后开始查找 SEH 例程入口（调试的情况除外）。首先就会找到与__try 块对应的__except 块。__except 的参数<exception>与 catch 的参数作用完全不同，也不类似于函数的参数，它主要是用于控制后面的程序执行，为这几个值之一：

EXCEPTION_EXECUTE_HANDLER（1），表示下面执行__except 块内及其后面的代码
EXCEPTION_CONTINUE_EXECUTION（-1），表示回到抛出异常处继续向下执行
EXCEPTION_CONTINUE_SEARCH（0），表示查找下一个异常处理例程入口

Microsoft 提供两个函数 GetExceptionCode(), GetExceptionInformation()，分别可以获取异常号和 EXCEPTION_POINTERS 类型的异常数据指针。而且这两个函数只能在__except 参数<exception>的表达式中使用。为了保证这一点，在 Microsoft Visual C++（以下简称 VC）中，编译器做了特殊处理，如果这两个函数没有在正确的位置，将产生编译错误。（这个感觉有点搞。）

所以，__except 一行一般会这样写：__except (ExceptFilterFunc(GetExceptionInformation()))，其中 ExceptFilterFunc 是一个自定义的异常处理例程，它输入一个 EXCEPTION_POINTERS *类型的参数，返回 EXCEPTION_EXECUTE_HANDLER、EXCEPTION_CONTINUE_EXECUTION 或 EXCEPTION_CONTINUE_SEARCH。

（注：下面所提到的“异常处理例程”，不管是自定义的还是系统提供的，都是这种类型的函数，这种函数指针类型在 winbase.h 中被定义为 LPTOP_LEVEL_EXCEPTION_FILTER。）

EXCEPTION_POINTERS 结构中包含丰富的异常相关数据，主要有异常号、异常发生时寄存器的值等。

与 try-catch 一样，__try-__except 也支持调用栈回溯，也可以嵌套，但没法重载。

另外，在 VC 中，还提供__try-__finally 块和__leave 关键字，这里不细说了，感兴趣的可以查查 MSDN。

2. Windows 异常处理步骤

回到上文的问题，没有匹配上 catch、__except 块的错误或异常将会如何处理呢？原来，包括__except 块在内，SEH 异常处理例程可以有多个，它们的入口地址形成一个链式结构，这个链式结构由 Windows 操作系统管理。

发生错误或异常后，Windows 的处理顺序一般如下：

（1）中止当前程序的执行。

（2）如果程序处于被调试状态，向调试器发送 EXCEPTION_DEBUG_EVENT 消息。

（3）如果程序没有被调试或者调试器未能处理异常，查找线程相关的异常处理例程（如对应__except 块）并处理。如果前面查找到的例程返回 EXCEPTION_CONTINUE_SEARCH，且线程有多个异常处理例程，则沿这些例程入口地址组成的链式

结构逐一向后查找，请求下一个例程处理。

（4）如果线程没有对应的异常处理例程，或线程所有例程都返回 EXCEPTION_CONTINUE_SEARCH，而且程序处于被调试状态，再次通知调试器。

（5）如果程序没有被调试或者调试器仍未处理异常，则进入主线程的“最终异常处理例程”链继续查找。

（6）“最终异常处理例程”链的最后是 Windows 默认的系统异常处理程序__CxxUnhandledExceptionFilter()，其处理通常是弹出一个异常对话框，上面显示一些异常信息，提供“关闭”、“调试”等按钮。

著名的 SetUnhandledExceptionFilter()函数就是在所谓“最终异常处理例程”链的__CxxUnhandledExceptionFilter()之前插入一个自定义的异常处理例程，当这个例程返回 EXCEPTION_EXECUTE_HANDLER 时，一般会直接结束进程。

三、两种异常处理机制的比较

我能想到的一些特征的比较：

	C++标准异常	SEH
局部对象析构函数	执行	局部对象有析构函数，且用__try-__exception 时，编译错误
可重载	有参数类型匹配	无条件处理
可移植	C++都有，不依赖操作系统平台	只有 Windows 提供
程序流程控制	catch 块后只能继续向下执行	EXCEPTION_EXECUTE_HANDLER、EXCEPTION_CONTINUE_EXECUTION、EXCEPTION_CONTINUE_SEARCH 三种流程控制，多个处理例程的依次处理
数据通用	各种不同的异常数据类型	统一结构的异常数据

四、VC 编译参数 EH

在 VC 中，你可能会发现一个怪异的现象，就是 try-catch 块无法捕获像“除 0”、“空指针访问”之类的异常。原来，在 VC 中一般的错误和异常都是用 SEH 来处理的，不等同于 throw 抛出的异常。而 try-catch 对结构化异常的处理，是由编译参数 EH 来控制的。

	无 EH 参数	EHs(EHsc)	EHa(同 Ehac)
try-catch	不处理异常	只处理 C++标准异常，代码优化较好	处理 C++标准异常和结构化异常，代码优化较差
__try-__except（VC2005 及以后）	处理 C++标准异常和结构化异常	处理 C++标准异常和结构化异常	处理 C++标准异常和结构化异常
__try-__except（VC2003 及以前）	只处理结构化异常	只处理结构化异常	只处理结构化异常

从表中可以看出，EH 参数对__try-__except 块的处理并无影响。

从 VC2005 开始，SEH 也可以统一捕获和处理 C++标准异常。而在 VC2003 及之前，C++标准异常只能由 catch 块来捕获。

VC2005 中，EH 参数默认为 EHsc。

附记：关于自定义 SEH 异常处例程的编写，如保存内存 dump，保存调用栈，使用调试相关的.pdb .map 文件等，网上相关的资料很多，需要可以查询。