

异常杀手-ExceptionalMagic

非法错误的杀手 - ExceptionalMagic

抱歉我用了一个耸人听闻的标题，不过我觉得 ExceptionalMagic 确实比我夸张的还要棒！那么什么是 ExceptionalMagic 呢？简单的说它就是一个内嵌到程序中的调试工具，当程序异常终止的时候，它会为我们提供一些非常有用的出错信息，使得我们可以轻松的定位错误。

众所周知，当我们的程序发生一些异常的错误的时候，Windows 也会为我们提供“非常详细非常底层”的错误信息，这就是著名的“非法存取错误... 位于\$abcdecf...”，我想如果我的客户告诉我他运行我的程序时碰到了这样的错误，并问我该如何解决的话，我一定是一边冒着冷汗，一边连忙回忆自己已经忘的一干二净的汇编知识，然后回答他：“这个~~~，让我想想~~~~~，哦，我想起来了，bill gates 一定知道这个错误是什么意思，等我打电话问问他后，再告诉你好吗？”。幸运的是当我拥有了 ExceptionalMagic 之后，我终于可以把 bill gates 抛到一边去了。ExceptionalMagic 这个魔术师可以告诉用户，错误发生的正确位置（源文件名，发生错误的代码行号，以及在错误发生前的一系列调用），你一定要问“really,这一切都是真的吗”，是的，我的回答是“of course ,I promise you”。

那么如何使用 ExceptionalMagic 呢？

首先，我们需要在单元中的 uses 部分添加对 ExcMagic 单元的引用，最好是把它放在所有其它单元的前面，这样的话 ExceptionalMagic 也可以处理其它单元初始化部分的错误了。

接下来，就是对 project options 选项做一些改变：打开 compiler 属性页的 stack frames 选项，并打开 Linker 属性页的 include TD32 debug info 选项（我们又一次体会到了这两个编译选项在调试程序时的强大威力了吧）。ExceptionalMagic 可以加载并使用连接到可执行文件中的 TurboDebugger 调试信息。编译程序后，你会发现程序几乎大了一倍，这时可以使用 ExceptionalMagic 带的 TDSPack 工具来压缩被附加的调试信息，并且这可以保证一些好奇的家伙无法使用 TurboDebugger 来察看我们的程序源码。

一个最简单的例子

先来创建一个新的项目，然后在单元中放置一个按钮，再在按钮的 OnClick 事件中添加下面代码：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    raise exception.create('demo');
end;
```

然后在单元的 uses 部分添加对 ExcMagic 单元的引用，运行程序，点击按钮，你会看到这回程序抛出的异常对话框已经被 ExceptionalMagic 替换成它自己的信息对话框了，信息里面包含了发生异常的单元名，代码行号，发生的过程名以及发生异常前的调用状况，如果你熟悉汇编的话，它还在 Register 属性页中提供了错误发生时的寄存器状态。

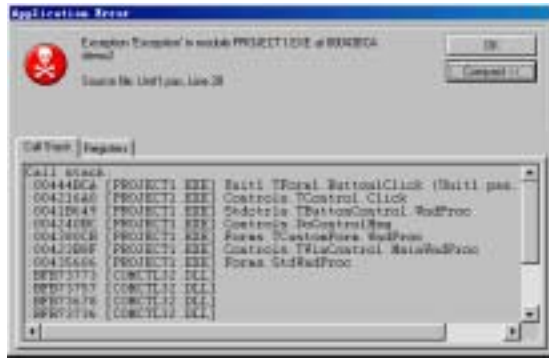


图 4.41

除此之外，ExceptionalMagic 还提供了一个 TExcMagicControl 控件，里面提供了很多错误信息对话框界面显示以及错误发送处理事件的控制选项，我们可以进一步定制，不过这里就不加介绍了，感兴趣的朋友可以自行钻研。

此外，前面提到了 TDSPack 工具，下面介绍一下它的用法。

用法:TDSPack [options] ExeFile [outfile]

下面是 Options 开关的列表:

表 4.1

-a	把调试信息附加在 EXE 文件后。
-e	从可执行文件中删除调试信息，并把它保存到独立的文件中。
-n	从调试信息中删除所有过程名，这将极大的减少调试信息的尺寸，剩余的调试信息中将只包含源码文件名和行号信息。注意：要和-o 开关一同使用才有效。
-o	优化调试信息(从 TD32 调试信息中删除所有 ExceptionalMagic 没有用到的信息，这会很有效地减少文件的尺寸)。

最后，要记住 ExceptionalMagic 的官方网址是 <http://dimus.virtualave.net/>。