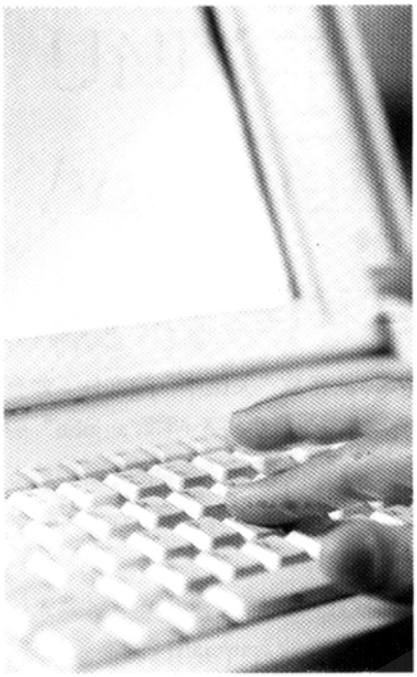

Delphi中异常处理技术探讨

李德军 （广东商学院电教中心 510320）

摘要：本文针对程序运行过程中的产生异常情况，运用Delphi开发工具并结合实例对异常产生的来源、异常处理的方法进行探讨。

关键词：Delphi 异常 被保护语句块 被保护资源块

1 引言

在程序运行过程中，往往出现一些意外错误状态而导至程序无法运行下去的情况，因此，在实现程序功能的同时，必须考虑如何处理程序运行过程中可能出现的各种异常情况。所谓异常是指打断程序正常流程的、非常见或意外的状态，打开文件时按路径找不到文件、不能分配所需的内存等。现本人结合实际就Delphi中异常的产生、来源以及实现异常处理的方法进行探讨。

2 异常的来源

异常可由一些不同的资源产生。用户的程序可以因为不正常状态产生异常。Delphi的组件可以因为不同的事件产生异常。把越界值赋给属性，或尝试为不存在的数组元素建立索引。运行的库过程和函数也能产生异常。执行带有非法操作的数学表达式，如被0除肯定是异常。其他产生异常操作的例子包括：引用nil指针，执行非法类型转换的表达式等。总的来说，软件在运行过程中遇到的情况千差万别，因此产生异常的情况也各有不同。

3 传统的异常处理方法

对于程序运行过程中产生的异常情况，传统处理的方法主要是通过IF语句、布尔型标志和特别函数返回值来控制。如在加载文件时，采用条件语句，以处理可能产生的打开空文件的错误。

```
If Opendialog.Filename=nil then //判断是否选择文件
Begin
  ShowMessage（“选择文件无效，请重新选择一个文件”）;
End;
```

以上例子是通过IF语句判断到异常事件后，提示重新操作，这种异常处理方法当然很容易理解，但在软件编写的过程中，如果每个地方都使用IF语句来检查错误并处理异常，会使编程工作变得较为繁杂。在Delphi中，这个问题可以用更简便的方法来实现异常处理。

4 Delphi中的异常处理方法

Delphi支持多种异常处理机制，Object Pascal提供了高级的异常处理机制。在这里就两种异常处理方法进行探讨。

4.1 创建被保护语句块

所谓被保护语句快，简单说是一条或多条语句，带有对这些语句产生异常的处理器。运用try和except关键字来创建被保护语句块的结构。具体结构如下：

```
try
  //可能引起异常的代码段
except
  //对异常进行处理的代码段
end;
```

其中try和except为关键字。Try用于标志可能产生异常的代码段，如果这段程序在运行时产生了异常，系统会中止try代码段的执行，并根据except所设的代码段进行。

下面就如何利用这个结构来创建被保护语句块研究一个程序的例子，具体的调用事件处理过程主要代码如下：

```
var
  I,J,K:Integer;
Begin
  I:=0;J:=10;
Try
```

```
K:=J div I; // 产生异常的语句
Except
    ShowMessage('Divide error!'+'I='+IntToStr(I)+'J='+IntToStr(J)+ 'K='+IntToStr(K));
end;
```

在这个过程中，表达式J div I尝试用10除以0，使Object Pascal运行库产生一个异常，从而执行了except块中的Showmessage语句。这个保护语句控制程序的执行过程，主要有以下三个要点：

（1）如果except块中的一条语句处理了一个异常，过程或函数在保护块后面继续正常运行；

（2）如果没有语句处理异常，当前的过程或函数立即退出，异常查找调用链，直到找到合适的处理器；

（3）未处理的异常最终找到应用程序的缺省异常处理器，处理器显示对话框和消息，缺省异常处理器接收所有没有提供处理的异常。

4.2 创建被保护资源块

以上的例子显示错误消息仅仅是异常处理的一个方面。当产生异常情况时，一个可靠的应用程序必须恢复稳定性。如果发生磁盘错误，应用程序必须释放可能不用的内存快，直到用户重新启动。一个可靠的程序很容易排除错误，通过关闭文件，释放Windows资源，用任何可能的方法从混乱中恢复。这种情况可用try和finally关键字来创建被保护资源块。所谓被保护资源块，简单说是由一条或多条语句组成，带有对资源使用可能产生异常的处理器。这个保护资源块的基本框架结构为：

```
try
 //程序语句
finally
 //释放的资源
end;
```

在这个结构中，不管try块中的语句是否产生异常，finally块中的语句总是要执行的。在典型的情况下，finally块中的语句释放内存，关闭文件、执行其他必须完成的操作来恢复系统稳定性。如果是try块外的语句产生异常，就会立刻引起过程或过程跳出，跳过finally块。需要注意的一点是：分配语句不要放在try块中，即使它将产生一个异常也是如此（如由于RAM的缺陷而使内存分配失误），finally块的目的是释放资源，因此，必须在try块前放置分配语句。在该块中，插入任何可以产生异常的语句，或可以让过程或函数退出以让分配的资源悬挂直到用户重新启动的语句。

下面结合实际例子来探讨这种技术，并且显示如何使用被保护资源块来防止悬挂资源。具体的过程主要代码如下：

```
var
  I，J，K：Integer;
  P：Pointer;
Begin
  I:=0; J:=10; //给变量初值
  GETmem(P,4098); //分配内存资源
  Try
    K:=J div I;
    ShowMessage('Results:' + 'I'+IntToStr(I)+ 'J'+IntToStr(J)+ 'KI'+IntToStr(K));
Finally
    FreeMem(P,4098);  //释放内存资源
    ShowMessage('已经释放内存');
End;
```

上述过程与被保护语句块相似，只是增加了一条分配内存的语句，这条语句调用GetMem来保存4098个字节的内存，当产生异常情况时，程序由于设置了被保护资源块，使得占用内存的资源得以释放。另外，程序中潜在的错误除法表达式存在于try块中，如果除法不在try中，这条语句的执行就将产生异常，过程会立刻结束。

5 结束语

在程序中加入异常处理结构以达到对异常的控制，这种处理方法在编写程序的过程中非常有用。有人可或许认为异常处理很复杂或者用处不大，其实异常处理并不复杂，无非是中断产生异常的程序代码，捕捉可能产生的异常并进行相应处理，掌握了这一基本方法后，就理解运用异常处理机制的重要性，以及如何使用异常处理机制了。■

参考文献

1 *Delphi4 实用大全／（美）斯旺（Swan，T）著；齐舒创作室译，北京：中国水利水电出版社，1999.1。*

2 *Delphi 高级开发指南 ［美］ Marco Cantu 等著，王辉等译，电子工业出版社。*