FORTRAN 2000异常处理机制的研究^{*}

鞠小林^{1,2,3} 徐宝文^{1,2}

(东南大学计算机科学与工程系 南京210096)¹ (江苏省软件质量研究所 南京210096)²
(南通工学院信息工程系 江苏南通226007)³

摘要 异常处理是现代程序设计语言的一个重要特征,它为检测和恢复软件系统在运行时的错误、构建系统的容错处理提供了强有力的支持。本文分析了 FORTRAN 2000 的异常处理机制,并提出一种扩充方案,引入用户定义异常和异常处理区的概念,改进了异常的传播机制,从而使 FORTRAN 编程更好地实现普通代码和异常处理代码的分离,提高了程序的可读性和可维护性。

关键词 FORTRAN, FORTRAN 2000, 程序设计语言, 异常处理, 健壮性

Exception Handling in FORTRAN 2000

JU Xiao-Lin^{1,2} XU Bao-wen¹

(Department of Computer Science & Engineering, Southeast University, Nanjing 210096)¹

(Jiangsu Institute of Software Quality, Nanjing 210096)²

(Department of Information Engineering, Nantong Institute of Technology, Nantong 226007)³

Abstract Exception handling, an important feature of modern programming languages, provides a strong support for detecting and recovering runtime errors, and structuring fault-tolerant activities in a system. This paper analyzes the exception handling mechanism of FORTRAN 2000, and proposes a extension through introducing user defined exceptions, exception handler and improving propagation mechanisms, thus make normal code and exception handle code separate better, and increase the readability and maintainability of programs.

Keywords FORTRAN, FORTRAN 2000, Programming language, Exception handling, Robustness

1. 引言

异常指引起程序正常执行挂起的事件,对此类事件的处理过程称为异常处理。一套有效的异常处理机制可以很好地检测程序运行时错误并使系统恢复到正常状态,从而构建系统的容错处理机制,使程序有较强的健壮性。随着软件系统的规模和复杂度的增加,越来越有必要在系统中增加异常处理代码以满足设计者对系统可维护性和可重用性的需求。据统计,在一般的软件系统中,大约有超过三分之二的程序代码是用于系统的异常处理^[1]。异常的检测和处理会影响程序的正常执行,降低程序的执行速度^[2]。如何设计简单、有效的异常处理机制是程序设计语言的研究重点之一。FORTRAN 是目前国际上广泛流行的一种高级语言,主要适用于科学计算。ISO/IEC/JTC1于2002年2月提出 FORTRAN 2000 草案,主要是在 FORTRAN 95 基础上的扩充。该草案增加了 IEC 60559 (IEEE 754) 异常处理功能^[2,3]。

本文在分析 FORTRAN 2000 的异常处理机制的基础上,提出一种异常处理的扩充方案:允许用户在特定的异常定义模块中定义自己的异常,改进了 FORTRAN 2000 的异常传播机制,并引入异常处理区的概念,程序执行时引发的异常,包括用户定义的异常均可以被处理器(processor)检测到,并在异常处理区内由相应的异常处理代码处理。这里的处理器是

指计算系统以及在此计算系统上的程序设计语言使用的转换机制的结合^[2]。该扩充方案较好地实现了程序中普通代码和异常处理代码的分离,可使程序的书写更为简洁灵活,对异常的处理更为可靠,提高了软件系统的可维护性和可重用性。

2. FORTRAN 2000的异常处理

异常处理机制一般包括异常的表示、引发、传播、处理、清除几个方面^[1]。本节首先介绍 FORTRAN 2000 中异常的表示,接着分析 FORTRAN 2000 的异常处理机制,并讨论处理器在停机模式下的异常处理情况,最后分析了此类异常处理机制的优点和不足之处。

2.1 异常表示

异常表示一般有三种方法:符号、数据对象与全对象^[1]。传统上异常用符号来表示,即用字符串或数字表示异常。FORTRAN 2000 支持 IEC 60559 定义的异常,用字符串来表示异常。每一异常有一个状态标记,可取异常引发和正常两种状态之一,初始状态为正常状态。IEC60559 定义了数据无效异常(IEEE_INVALID)、数据上溢异常(IEEE_OVERFLOW)、除数为零异常(IEEE_DIVIDE_BY_ZERO)、数据下溢异常(IEEE_UNDERFLOW)以及数据不精确异常(IEEE_INEXACT)等五种异常^[2]。

FORTRAN 2000 不要求处理器一定支持 IEC 60559 定义

^{*} 本文部分得到国家自然科学基金项目、国防“九五”及“十五”重点预研项目、国防武器装备预研基金项目、江苏省自然科学基金项目、江苏省科技攻关项目、武汉大学软件工程国家重点实验室开放基金项目、江苏省计算机信息处理技术重点实验室开放基金项目(苏州大学)的资助。鞠小林 硕士研究生,徐宝文 教授,博士生导师,主要从事程序设计语言、软件工程、并行与网络软件等方向的教学与科研工作。

的所有异常^[2]。对于处理器不支持的异常,其状态标记始终为正常状态。

2.2 异常处理机制

异常处理机制的核心包括异常的引发、传播与处理三个方面^[4]。

·异常引发 异常的引发机制通常有两种:(1)基于程序设计语言的机制:用程序设计语言来定义并且通过这种语言的编译程序来实现异常的引发;(2)基于操作系统的机制:使用操作系统提供的库函数或系统调用来实现异常的引发。FORTRAN 2000同时支持这两种异常引发机制:数据上溢、数据下溢、除数为零这些异常的引发基于操作系统机制;数据无效、数据不精确这两个异常的引发基于程序设计语言机制。在程序执行的过程中,如果发生了某个异常,则处理器将其对应的异常状态标记置为异常引发状态。

FORTRAN 2000草案分别按照如下方式引发这五种异常^[2]:对数据上溢异常,当对实数进行运算或者赋值操作,或者对复数的实部或虚部进行运算或者赋值操作,若其结果的绝对值大于处理器支持的数据的表示范围的上限,引发数据上溢异常;对数据下溢异常,当对实数进行运算或者赋值操作,或者对复数的实部或虚部进行运算或者赋值操作,若其结果的绝对值小于处理器支持的数据的表示范围的下限,引发数据下溢异常;对除数为零异常,当对实数或者复数做除法运算,如果分子不等于零,而分母等于零,则引发除数为零异常;对数据无效异常,当对实数或者复数做运算或者赋值操作无效时,引发数据无效异常;对数据不精确异常,当对一个实数或者复数做运算或者赋值操作时,其结果不精确,则引发数据不精确异常。当上述异常被引发时候,对应的状态标记 FLAG 的值变为异常引发状态(SIGNALING)。模块 IEEE_EXCEPTIONS中提供了对异常状态的设置和查询功能。

·异常传播 异常传播机制规定了在发生某个异常时处理器如何找到相应的异常处理程序^[4]。如果在异常处理区中不存在对引发的异常的处理代码,则称异常被传播了^[6]。FORTRAN 2000中采用了简单的异常传播机制:简单地传递异常状态标记,把异常状态标记作为全局变量,当异常引发后,异常的状态标记被设置为异常引发状态,而程序继续按顺序执行,直到遇到相应的异常处理程序,异常处理程序通过对异常状态标记的判断决定是否执行异常处理程序。

FORTRAN 2000关于异常的状态标记,对处理器作了如下四点规定^[2]:

(1)若一个域单元不支持 IEC 60559定义的异常,则该域单元对异常处理的支持就依赖于处理器。程序在调用这类域单元的时候,当某一异常状态标记为异常引发状态时,在退出该单元时保持该异常的标记不变;当异常状态标记为正常状态时,在返回时异常状态标记是否改变取决于处理器的异常处理机制。

(2)在进入一个过程时,若某个异常状态标记为异常引发状态,则处理器在过程入口处将异常状态标记置为正常状态,并且在返回时恢复为原状态。

(3)在一个过程执行中,若引发了某个异常,则将该异常的状态标记为异常引发状态,在过程返回时,其异常状态标记保持不变,除非在程序中使用改变异常状态标记的过程将其状态改为正常状态。

(4)在未调用查询异常状态的函数和设置异常状态的过程的语句序列中,若其中某一操作可能引发异常,而该操作不

影响这一语句序列之后的变量的值,则由处理器的异常处理机制决定是否抛出异常。

·异常处理 异常处理涉及两个方面:异常处理区的位置与异常处理模型。FORTRAN 2000没有具体规定异常处理区的位置。可以采取随文处理法,即什么地方发生异常就在什么地方处理;也可采用在每个过程的最后加上相应的异常处理程序,甚至可以对异常不作处理。通常的异常处理如图1所示。
CALL IEEE_GET_FLAG(异常名,异常引发状态)!查询异常是否被引发
IF (异常引发状态) THEN
!异常处理部分
...
END IF

图1 FORTRAN 2000的异常处理

我们可以通过查询异常状态标记来设计异常处理代码。如果要处理多个异常,则可以采用多个这样的条件判断语句组合运用。

异常处理模型有多种,一般有中止模型、重试模型、恢复模型等等^[1-4]。FORTRAN 2000对异常处理模型并不做具体的规定。可以采用中止模型,即在异常引发后,执行到异常处理区,在执行完时不返回到异常引发点;也可以采用重试模型,即可以在异常处理结束时将控制转移到异常引发点。重试模型在异常处理程序中必须有异常状态标记的清除语句,而且控制转移也要由用户来实现,重试模型比较适用于随文处理法。

2.3 停机处理

停机处理指当程序在执行过程中发生异常时,处理器可以控制程序中止或继续运行。在 FORTRAN 2000中,可以进行函数查询或者修改停机功能的设置。若处理器支持,允许通过调用过程设置程序在某个异常引发后是中止还是继续执行。在调用过程的入口处,处理器不改变停机模式,并且保证在返回时停机模式和入口处一样。

2.4 FORTRAN 2000异常处理机制的优点与不足

通过以上分析可知,FORTRAN 2000对异常类型、异常状态标记、异常状态操作等在一个模块中给出定义和声明,使用时只需通过 USE 语句引入,实现了信息的良好封装性;通过异常状态标记实现异常的传播,简化了 FORTRAN 语言的异常处理机制的设计;异常处理可以出现在异常引发点之后的任意地方,异常的处理比较灵活。但与 Ada、C++、Java 等程序设计语言相比,它的异常处理机制还不是很完善,存在很多不足,表现如下:

(1)支持的异常类型很少,支持 IEC 60559定义的五种异常,不支持队列、堆栈、数组溢出、超时、资源冲突等异常,对异常的处理能力十分有限。

(2)异常的传播机制很不完善,异常引发后,程序的控制流不能立即转到异常处理程序,而是继续按顺序执行,直到执行到异常处理程序。由于在很多情况下无法确定异常引发的位置,采用在哪里发生异常哪里处理的随文处理法很难实现。这种传播机制可能使异常不能得到及时的处理,从而导致更多的异常发生。

(3)不支持异常状态标记的自动清除。在异常处理后,异常状态标记仍旧为异常引发状态,需要显式恢复为正常状态。

(4)对异常的处理的位置比较随意,只要在异常的引发点之后就行。这种方式容易导致异常处理的代码比较分散,而且异常处理要通过先查询异常状态标记判断异常是否发生,再

进行相应的异常处理,使程序十分繁琐,从而降低了程序的可读性。

3. 一种异常处理扩充方案

针对 FORTRAN 2000 异常处理机制的不足之处,我们提出一种异常处理的扩充方案:引入用户定义异常,改进 FORTRAN 2000 的异常传播机制,并且引入异常处理区的概念。在发生异常时,就把控制转移到异常处理区中相应的异常处理程序,但结束后不返回原来的异常引发点。这种方案增加了对用户定义异常处理的支持,丰富了异常处理类型,将异常处理代码采取集中式的处理方式,可以使编写的程序简洁,层次十分清晰,可读性大大增强。

3.1 用户定义异常的表示及引发

在 FORTRAN 2000 中,异常被预先定义在模块 IEEE_EXCEPTIONS 中,用户通过 USE 语句加以引用。为此,我们增加一个用户定义异常模块 IEEE_USERDEFINED_EXCEPTIONS,将程序中使用到的用户异常在该模块中预先定义。用户定义的异常用 FORTRAN 标识符表示,且不能和 IEEE 60559 定义的五种异常名以及 FORTRAN 语言的关键字相同。用户定义的异常在用户定义异常模块中直接可见,对于通过 USE 子句引入用户定义异常模块的域单元可见,对其它域单元均不可见。

对于用户定义的异常引发,在用户定义的异常模块中给出相应的操作过程(IEEE_SET_FLAG)用于设置异常的状态标记,可以通过调用该过程抛出用户异常。

3.2 扩充后的异常的传播机制

在软件系统中,异常的传播一般有显式传播与隐式传播(也叫自动传播)两种解决方案^[1-7]。按照隐式传播方式扩充 FORTRAN 2000 的异常传播机制。处理器发现异常后自动将控制转到异常处理模块。若本层次没有相应的异常处理模块或者在执行异常处理模块时发生了异常,则要向上一层模块传播异常。若引发的是用户定义的异常且向上传播时上一层次没有引入用户定义异常模块,则直接调用上一层次的缺省异常处理程序;若无缺省异常处理程序,则继续向上传播;若到了最外层模块后仍没有相应异常处理程序,则程序运行终止或者使用系统的缺省处理程序。

3.3 扩充后的异常处理语法

参照 CLU、Java 和 Ada 的异常处理语法^[1-6,8],我们引入 EXCEPTION 关键字,图 2 给出了我们扩充的 FORTRAN 2000 异常处理语法:

```
EXCEPTION WHEN 异常名1:异常处理块1
             WHEN 异常名2:异常处理块2
             ...
             WHEN 异常名 n:异常处理块 n
             [DEFAULT: 异常处理块 n+1]
END EXCEPTION
```

图 2 扩充后的异常处理区

异常处理区是位于关键字 EXCEPTION 和 END EXC-

PTION 之间部分。其中,异常名为模块 IEEE_USERDEFINED_EXCEPTIONS 中预先定义的异常,也可以是 IEC 60559 定义的五种异常。WHEN 语句表示,当异常 i 的状态标记为异常引发状态时,执行异常处理块 i,并将异常状态标记恢复为正常状态。DEFAULT 为可选项,用于处理异常名 [1...n] 之外的异常。

异常处理区负责处理在异常处理区之前引发的异常,通常安排在过程的最后。异常处理区和异常引发点采用静态捆绑方法^[1],程序执行时,一旦有异常被引发,处理器将控制转到异常处理区,并寻找对应的异常处理程序,如果没有对应的异常处理程序,则向上一层传播。采用这种方案,我们可以将普通代码和异常处理代码分开来,提高程序的可维护性和可理解性。

结束语 本文在分析 FORTRAN 2000 的异常处理机制及其不足之处的基础上,借鉴 CLU 和 Ada 等语言的异常处理机制,提出了一种对其异常处理机制的扩充方法:引入用户定义异常和异常处理区概念,改进 FORTRAN 2000 的异常传播机制,从而使 FORTRAN 编程更加灵活,更好地实现普通代码和异常处理代码的分离,提高了程序的可读性和健壮性。当然,FORTRAN 2000 语言的异常处理机制还不是很完善,在异常处理的动态绑定、处理并发异常、异常的清除等方面,还有待于我们进一步研究。

参考文献

- Garcia A F, Rubira C M F, Romanovsky A, et al. A comparative study of exception handling mechanisms for building dependable object-oriented software. *The Journal of Systems and Software*, 2001, 59: 197~222
- J3 (2002)J3/02-007R3 - Draft FORTRAN standard. Also known as ISO/IEC JTC1/SC22/WG5 N1497. Available as PS, PDF, or text from ftp://ftp.j3-FORTRAN.org/j3/doc/standing/2002/02-007r3/John Reid, WG5
- Convener. The New Features of FORTRAN 2000. Also known as ISO/IEC JTC1/SC22/WG5 N1495
- Buhr P A, Mok W Y R. Advance Exception Handling Mechanisms. *IEEE Transactions on Software Engineering*, 2000, 26(9): 820~836
- Garcia A F, BEDER D M, Rubira M F. An Exception Handling Mechanism for Developing Dependable Object-Oriented Software Based on a Meta-Level Approach. In: Proc. of the 10th IEEE Intl. Symposium on Software Reliability Engineering, 1999
- Romanovsky A. Extending conventional languages by distributed/concurrent exception resolution. *Journal of Systems Architecture*, 2000, 46: 79~95
- 陈娟,张素琴,王萍. 程序设计语言异常处理机制. *微型计算机与应用*, 1998, 10: 15~17
- 徐宝文. 高级程序设计语言原理. 北京: 航天工业出版社, 1992