

C - Java 自动程序转换系统的设计

严忠林 张辅群 徐剑峰

(上海师范大学 数理信息学院 上海 200234)

摘 要 程序设计语言的相互转换技术可以被广泛运用在软件维护、遗留系统的升级改造以及软件逆向工程等领域中。文中先对现有的几种移植方法进行了分析和研究,分析表明在将程序库移植到 Java 中和将它们与 Java 整合时,这些方法暴露出了各自的局限性和不足。借鉴语言转换经验,制定了转换的设计原则并探讨了将 C 语言转换到 Java 语言的过程中需要解决的一些问题,以及这个转换系统的设计思想和实现方法。文中所阐述的内容为实现异种程序设计语言的程序代码转换,提高程序代码的可移植性和重用性提供了有意义的思路和实现方法。

关键词 程序语言转换 转换系统 指针 抽象语法树

中图分类号 TP311

文献标识码 A

文章编号 1673-629X(2007)02-0046-04

Design of C to Java Automatic Transformation System

YAN Zhong-lin, ZHANG Fu-qun, XU Jian-feng

(Mathematics & Science College, Shanghai Normal University, Shanghai 200234, China)

Abstract The transformation technology of programming languages can be widely used in software maintenance, updating of legacy systems and software reengineering etc. In this dissertation, survey and evaluate current approaches to the migration of source code to Java. The survey of current migration approaches reveals a number of their restrictions and disadvantages in the context of moving program libraries to Java and integrating them with Java programs. Using the experiences from language transformation survey, established a number of goals for an improved translation approach and discussed some critical problems need to be solved in transformation from C to Java language, and then introduces the design idea and implementation method of this automatic transformation system. The contents of this paper on some consideration and effective methods of carrying out the transformation between the different programming languages' source codes to improve these code's capabilities of transplant and reuse.

Key words programming language transformation, transformation system, pointer, abstract syntax tree

0 引言

程序转换是指从一种高级程序设计语言到另一种高级程序设计语言之间的转换,这是在源代码间的转换。在软件移植、重用、更新、编译等方面使用自动转换技术,可节省大量的软件开发费用,缩短开发周期,因而受到业界的广泛关注。

以往研究的程序转换技术大多着眼于从过程式语言到过程式语言。现在由于面向对象的语言,特别是跨平台的 Java 语言的广泛使用,选择经典的过程式语言 C 到 Java 语言的转换作为专业人员的研究课题则是顺理成章的。

1 相关工作

目前,国外所做的许多工作是关于 Java 到 C 的转换或优化,如 Harissa^[1], j2c^[2], JCC^[3]和 Tobá^[4]。而从事 C 到 Java 的研究很少,主要相关的有 c2j^[5]和 c2j++^[6],都是 C++ 到 Java 的转换工具。c2j++ 是基于 c2j 的,主要的不同是 c2j++ 是用 Java 写的,而 c2j 是用 C 写的。不幸的是这些工具都有明显的局限性。它们基本上都忽略了指针的问题,这是 C 到 Java 转换的主要难点。这些工具简单地移去所有指针的取值运算符,也就是移走星号和将“→”转换为“.”。当处理对象和指向对象的指针时,这种做法通常有效,但如果是对数组就没有作用。对科学计算软件来说,数组是最重要的通用数据结构,因此它们的转换是不可忽略的。

其它相关的工作是一个 Fortran 到 Java 的转换器 f2j^[7]。有意思的是,它把 C 作为中间语言表示,也就是它将 Fortran 代码转到 C,再将结果转换到 Java。然而,它这样做纯粹是为了方便,因为 Fortran 到 C 的转

收稿日期 2006-05-07

基金项目:上海市教委基金项目(05DZ14)

作者简介:严忠林(1960-),男,江苏镇江人,讲师,研究方向为计算机系统、程序设计等。

换器已经有了,但它并不支持通用 C 到 Java 的转换,尤其是不能对指针进行处理。

2 C 到 Java 的移植策略分析

目前,将 C 代码移植到 Java 中有几种不同的解决方法:可以用 Java 来包装 C 代码,将其作为本地二进制代码调用;可以把 C 代码编译转换为 Java 的字节码,再和其它 Java 程序整合;可以用 Java 来重新设计和实现整个项目工程;还可以选用一个源代码自动转换系统来实现。

2.1 本地代码

使用 Java 提供的 JNI 技术将 C 代码整合到 Java 中,这种方法易于实现且对 C 源代码的修改也较少,但是必须提供接口类来实现 C 和 Java 的通信。由此产生了大量的接口,更重要的是这些接口要进行频繁的数据类型转换,从而制约了运行时的性能。用户不得不权衡用 C 获得的性能提升和接口导致的性能损失。

这种方法主要缺点是由于使用了 JNI 技术而丧失了 Java 语言特有的平台无关性。C 的本地代码是针对特定平台编译生成的,C 和 Java 的混合语言就只能在这一特定平台上运行。这种方法适合于那些只运行在专门平台上的服务器端程序,但不适于通常需要运行在多种不同平台上的客户端应用程序。

而且,由于 C 代码不在 Java 虚拟机的安全环境中执行,因而绕开了 Java 的安全检查机制,带来了无法预料的安全隐患。

2.2 Java 字节码

编译器“*The Java Back-End for GCC*”^[8]可以实现从 C 编译器产生 Java 虚拟机的字节码。这看上去像个好策略,然而,为了将 C 运行时环境的内存处理正确地映射到 Java 虚拟机中,以及可以在 JVM 中进行类型检查,不得不使用一个巨大的数组来存储 C 源代码中所有的变量。如果一个 C 程序有缺陷将会导致整个数组的崩溃,而且错误也无法预料和很难诊断。同样,这种方法也需要专门的接口来处理 Java 代码和 C 代码之间的通信,存在上一种方法的缺点。

2.3 重新实现

另外一种可行的方法是使用原有的代码文档,遵循正向软件工程方法来重新设计和实现代码。然而,在实际中有很多案例表明这些文档或者缺失,或者不能反映系统的真实结构和功能。而且往往由于时间上的压力,设计文档通常不能和代码同步,这样需要通过软件逆工程的方法从源代码中提取设计,然后再形成设计文档,根据目标语言的特性来改变设计,进行分

析、调整和优化后再以目标语言实现。这种方法适合于当前系统存在维护上或性能上的问题,或有较大的需求改动。采用这种方法需要太多的人工干预。

由于原本是寻求一种能够对大量源代码进行自动转换的方法,而不需更改源代码的设计,所以不采用这种方法。

2.4 源代码转换

前两种方法对原始的代码改动较小,在项目移植前期快速方便,但在项目后期会给系统维护和功能重用带来极大的困难。重新设计和实现可以得到最佳的 Java 代码,但它的费用高昂,开发周期长。而如果使用自动转换系统可以得到较高的性价比和良好的维护性能,那么这就成为了目前可行的较好的一种选择。当然,除此以外还需要进行测试来消除转换后系统和原系统存在的差异。

3 转换前需要考虑的因素

C 和 Java 的不同主要是由于编程语言的设计目标不同造成的,C 是用于支持底层和高性能的系统开发,而 Java 的重点是安全和网络应用。尽管它们有相似之处,但在实际编程时,它们使用不同的设计理念。

一方面 Java 有许多优于 C 之处,如自动垃圾回收、边界检测、丰富的标准库、包,最重要的是跨平台性和易操作性;另一方面,由于结构的变化,如指针、位字段和内存管理等,Java 可能不是底层程序的理想选择,如嵌入式系统、硬件驱动或时间要求严格的软件。所以若希望转换到 Java,需要考虑如下因素:

(1) 要移植原系统的哪部分?如果是完全移植,重新设计比坚持原始的设计更好。如果是部分移植,则会遇到上面所介绍的一些问题。如果你的目标是使软件的一个版本能够在所有的环境里编译执行,那这种移植就是值得做的。

(2) 如果原系统非常注重效率,那么不要移植。因为 Java 在性能上要比 C 差,所以性能下降是可预见的。

因此,在对其进行转换前需评价源系统对自动转换的适宜性。如果代码使用了大量 Java 没有的语言特性,那么应该选择手动转换,因为对这样的代码没有一个自动转换是安全有效的^[9]。

由上所述,假设这些代码对自动转换是适宜可行的,还假设 C 代码是正确有效的。如果源程序代码本身是错误的,那么就很难实现对它的正确转换,实现转换后的目标程序代码也难以正确执行。所以这就要求系统的输入其本身在词法、语法和语义这三个方面必须是正确的。

4 转换系统的设计

4.1 转换系统的设计原则

大量的实践经验表明^[9],在语言转换过程中需注意如下一些重要的影响因素:

(1) 语言转换通常是在重要商业系统的关键部分上,所以重点必须放在转换过程的可靠性上。

(2) 执行转换的软件工程师必须是源语言和目标语言的专家,并且能意识到在不同语言之间转换可能出现的复杂问题。

(3) 转换后的系统不一定要和用目标语言专门开发的系统具有同样的设计。

(4) 转换一个语言特性丰富的源语言到语言特性匮乏的目标语言是相当困难的,甚至是不可能进行的。

(5) 转换后的系统不应比原系统太大。

理想上,转换后的系统应该是使用目标语言的语言特性和编程习惯,事实上产生的目标系统中还是保留了源语言的方言。使用目标语言的语言特性可以使目标系统易于维护,但显然会降低转换的自动化程度,也影响转换结果的语义正确性。而对非内置的语言结构的模拟可以提高转换的自动化程度,得到语义正确的程序,但会使新系统的二次开发和维护变得复杂,也会降低系统的运行性能。因此,需仔细权衡利弊,一般情况下应尽可能地使用内置的语言结构,对语言结构的模拟往往会对整个转换工程产生较大的负面影响^[9]。

根据前面讲述的转换经验,对于所要进行的转换工作应遵循以下几个原则:

a. 首要的原则是应该达到最大的自动化程度。虽然有一部分仍要依靠程序员手动干预,但转换系统将对这些问题产生建议来指导他们解决问题。

b. 在转换过程中,应保留源代码的逻辑结构,保持源程序的各种信息能较完整地反映到目标程序中去。尽量使用 Java 中的语言特性,减少对 C 语言特性的模拟,提高目标系统的可读性,使产生的代码具有良好的维护性。

c. 产生的代码应最大程度地等价于原来的代码,保持转换前后的程序代码在执行流程(语义流程、逻辑结构)上的一致性。

d. 产生的代码的性能不应比由程序员开发的代码的性能差太多。

4.2 转换规则的分类设计

根据设计原则,将 C 语言到 Java 语言的转换分为下列三种关系:

(1) 简单兼容。

这种代码转换比较简单,基本上可以一一对应。

不需处理或稍加处理即可进行自动转换。这时可以将 C 程序转换为语法和语义上完全等价程序。这类转换是安全的。

(2) 复杂兼容。

不能直接对应,需加额外的处理或生成具有相同功能的 Java 代码段。在大多数输入的情况下,经过自动转换后目标程序与源程序是等价的,但是并不能保证所有的输入情况均等价。这类转换被认为是不安全的。处理方法是在转换后提示用户存在这样的不安全性。

(3) 不兼容。

很难找到与之对应的转换规则,难以生成具有相同功能的 Java 代码。这时如果进行自动转换将带来极大的负面影响,故一般不对其进行转换,建议用户自行修改。

5 转换系统的实现方法

5.1 系统实现的基本思想

可以将 C 到 Java 语言的转换看作是编译技术中的语义处理部分。通常有两种方法进行语义处理:一是语义处理作为独立任务,它以语法分析得到的中间结果(源程序的内部表示)作为输入,进行语义操作的处理。二是采用语法制导的方式,即在语法分析用到的产生式中插入语义动作,在语法分析的同时执行语义代码。文中采用第二种方法,使用自动生成工具产生词法和语法分析器,并在生成的语法分析器中嵌入语义动作,实现 C 到 Java 的转换。

5.2 系统的体系结构

根据转换的需求,系统需要进行的主要工作是:首先对 C 源程序进行词法和语法分析,然后对语法分析所收集到的相关信息进行分析,遵循所设计的转换规则进行相应的处理。转换系统的体系结构如图 1 所示。

(1) 预处理。

转换系统的预处理部分主要完成对源程序进行逻辑分段和与 C 预处理器的类似的功能。逻辑分段的功能是将源程序分解为逻辑功能独立、代码量较小的代码段,使得转换系统其后的部分能有效地进行分析和处理。

由于 C 是面向过程的设计语言,如何将其代码分成一个个独立的逻辑类是亟待解决的难题。到目前为止,还没有一个有效的算法或方法能够对程序源代码进行有效的逻辑功能划分,所以预处理部分中的逻辑分段工作大多是以手工的方式进行的。由于 C 源程序是由多个文件组成,大多数情况下可以把每个单独

的物理文件视为一个独立的逻辑功能代码段。

(2) 词法分析程序。

其功能是遵照源程序设计语言的语法规则定义，识别出源程序段中的单词符号(记号)，进入词法判定过程。记号要经过类似于传统编译系统进行词法分析，连同经过词法分析、扫描处理后产生的相关信息存储到记号词典中。

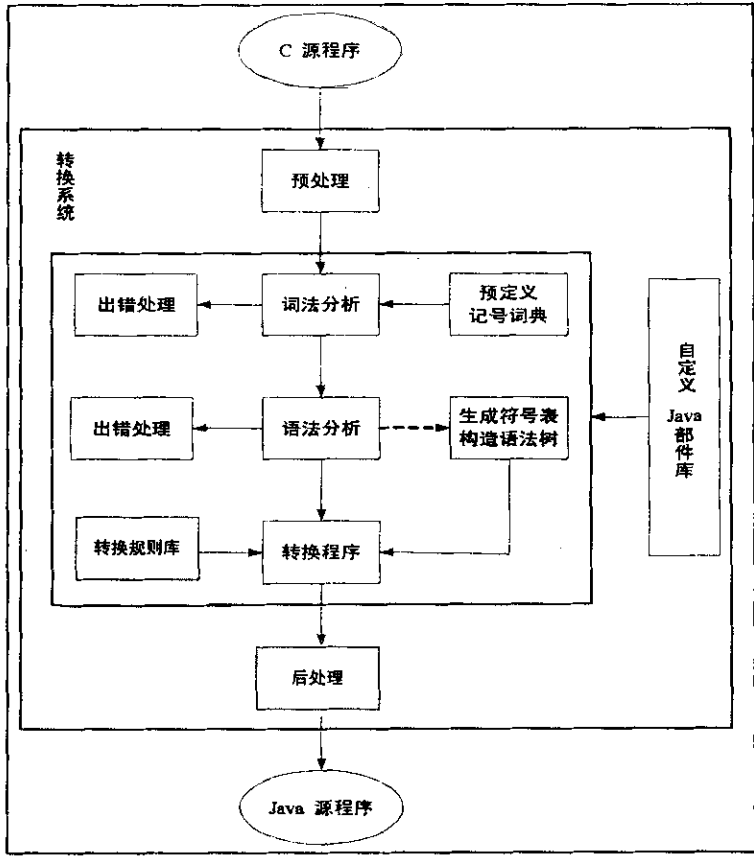


图 1 转换系统的体系结构

(3) 记号词典。

记号词典是已经具有源程序段记号数据的词典文件。记号词典给出源程序段中记号的目标程序的描述，并将这些数据对应地存储入记号词典文件中，供转换程序使用。

(4) 语法分析程序。

其功能是遵照源程序设计语言的语法规则定义，对源程序段进行语法分析，识别出源程序段所具有的语法结构，并将该语法结构用语法树的形式描述出来，为转换系统下一阶段提供进行分析处理的基本信息。

(5) 符号表。

符号表是系统进行翻译转换的基础，系统其后的部分在进行分析处理时，将会对符号表进行大量的查询和写入的动作。符号表采用了线性链表这种数据结构来记录数据信息。文中设计的符号表是对传统的编译器中符号表的扩展，其结构如图 2 所示。

符号表编号	源记号类型	源记号值	源记号属性域	源记号关联域	目标记号类型	目标记号值	目标记号属性域	目标记号项	目标记号值代码段
-------	-------	------	--------	--------	--------	-------	---------	-------	----------

图 2 符号表

说明 符号项序号表示该表项中记录的记号来自于哪个源程序段。源记号类型、源记号值和源记号属性域分别表示了该记号属于何种类型以及具体的属性值。

例如标识符 `abcd` 作为记号存储在记号词典中，那么它的源记号类型项的内容就为 `ID`，而源记号值的内容就是串值 `abcd`，如果该标识符具有 `real` 类型的属性值，则源记号属性域的内容就为 `real`。目标记号类型、目标记号值和目标记号属性域分别表示了用目标程序设计语言描述的对应记号的信息。有时一个源程序设计语言的记号，无法用一个目标程序设计语言的单词符号对应地表示出来，而是需要一段目标程序设计语言的程序代码来进行对应地表示，这段程序代码就需要存储于记号词典的目标记号值代码段这一项中，并用目标记号项加以指出。

对符号表最频繁的操作是建立新项和搜索旧项。

(6) 转换程序。

其功能是依照转换系统前面部分所获得的源程序段逻辑结构，对源程序段的记号进行相应地分析和辨识，进而从符号表文件中找到该记号对应的目标程序设计语言描述，按照一定的算法进行代码的组织和处理，最终获得用目标程序设计语言描述的程序代码段。

(7) 自定义 Java 部件库。

尽可能地找到对应的 Java 库函数，使其与 C 的库函数具有相同功能。如果找不到相同的库函数，需理解 C 库函数的具体含义后，用 Java 语言编写能实现相同功能的部件子程序，加入自定义 Java 部件库中。

6 结束语

由于这两种程序设计语言之间的差异，如语言设计典范、语法细节以及类库支持等，这种转化还需要处理很多问题，这有待进一步深入的研究。

文中阐述的内容为提高软件的可移植性和重用性提供了有意义的思路和实现方法，同时也适合于其它不同程序设计语言的之间的相互转换。

景反走样、深度检测等功能,实现对造型体运动以及对场景的真实模拟。

4.3 导航窗口和可变视点设计

导航窗口及多视图,通过在同一个窗口里划分多个子窗口,为视图创造分频效果(见图 1 下)。具体实现是运用 OpenGL 的 `glViewport` 函数实现分频,目的是在具体观察地球、月球或卫星的时候能对卫星空间态势同时有一个了解^[5]。

视点变化有两方面的含义:一是视点与场景相对位置的变化,即从不同方向观察卫星。本系统设计了用鼠标和键盘分别实现视点的任意变化,用以观察整个空间态势;二是视点随动,因为地球处于自转状态,若要观察卫星扫描区域必须使视点随卫星运动。为解决这一问题可将视点设计成追踪卫星,由 `GlutLookAt` 函数实现。系统仿真设计结果见图 1。

5 结 论

卫星在发射及在轨道运行时需要监测的数据量很大,理论上卫星监测图形软件可以做到对所有星上设备进行图形显示监测,但考虑到软件工作量、目前计算机的运行速度及监测范围的要求,目前的软件显示部分只包含卫星的太空位置、卫星的姿态、天线和帆板状态及部分关键部件的可视化。要开发逼真的三维仿真程序,还需研究许多具体问题,如对象的数据结构、模型数据库、可视化建模方法、渲染效率、仿真视觉效果等。卫星三维视觉仿真对于卫星检测系统和卫星管理系统具有极大的实用价值。

参考文献:

- [1] 吴 斌,段海波,薛凤武. OpenGL 编程权威指南[M]. 北京:中国电力出版社,2003.

(上接第 49 页)

参考文献:

- [1] Muller G, Moura B, Bellard F et al. Harissa: A flexible and efficient Java environment mixing bytecode and compiled code [C]//in Proceedings of the 3rd Conference on Object-Oriented Technologies and Systems. Berkeley, France: University of Rennes, 1997: 1-20.
- [2] Andoh Y. j2c/CafeBab[EB/OL]. 1998. <http://www.webcity.co.jp/info/andoh/java/j2c.html>.
- [3] Shaylor N. JCC - A Java to C converter[EB/OL]. 1997. <http://compilers.iecc.com/comparch/article/97-05-149>.
- [4] Proebsting T A, Townsend G, Bridges P et al. Toba: A Java to C translator[EB/OL]. 1999. <http://www.cs.arizona.edu/sumatra/toba/>.

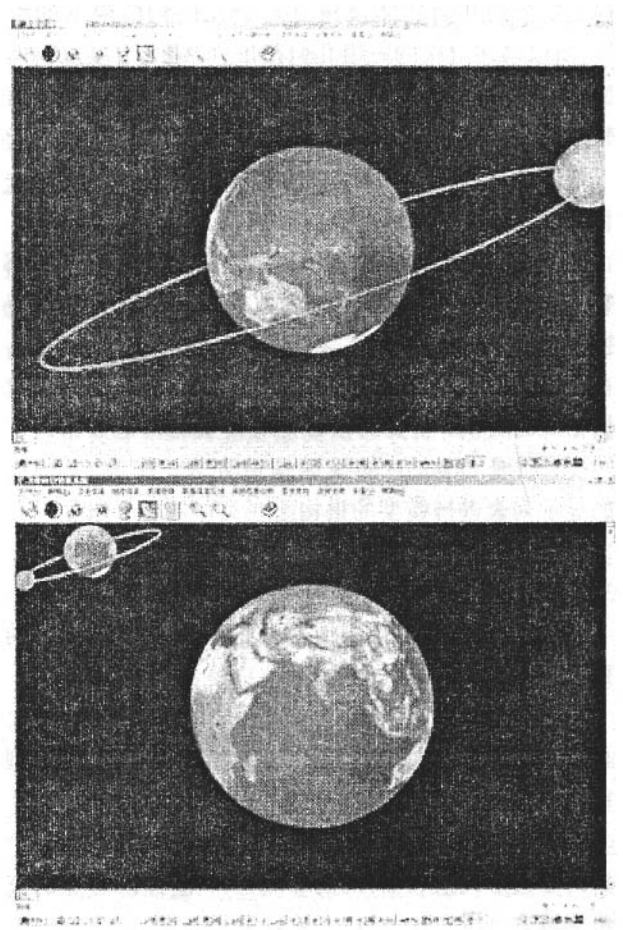


图 1 卫星跟踪仿真结果

- [2] 惠晓钟. 3DStudioMAX 新创意[M]. 西安:西安电子科技大学出版社,1999.
- [3] 廖朵朵,张华军. OpenGL 三维图形程序设计[M]. 北京:星球地图出版社,1996.
- [4] 李 颖,薛海斌. OpenGL 技术应用实例精粹[M]. 北京:国防工业出版社,2001:127-129.
- [5] Neider J. OpenGL Programming Guide[M]. [s.l.]: Addison Wesley, 1993: 42-56.

- [5] Laffra C. c2j. A C++ to Java translator[EB/OL]. 2001. <http://www.novosoft.us/solutions/tools.shtml>.
- [6] Tilevich I. C2j++[EB/OL]. 1999. <http://www.vbasicmaster.com/html/c2j++.html>.
- [7] Fox G, Li Xiaoming, Zheng Qiang et al. A prototype of FORTRAN-to-Java converter[C]//in Proceedings of the ACM 1997 Workshop on Java for Science and Engineering Computation. Las Vegas, NV: Syracuse University, 1997.
- [8] The UQBT team. Java Backend for GCC[EB/OL]. 2001. <http://www.itee.uq.edu.au/cristina/uqbt.html#gcc-jvm>.
- [9] Harsu M. Re-Engineering legacy software through language conversion[R]. A-2000-8. Tampere: Department of Computer and Information Sciences, University of Tampere, 2000.