基于 Python 的混合语言编程及其实现

罗 霄 任 勇 山秀明
(清华大学电子工程系 北京 100084)

摘 要 Python 是一种被广泛使用的脚本语言,它特别适用于混合语言编程的软件开发。本文介绍了 Python 语言的特点,给出了混合运用 Python 语言和系统编程语言开发的软件结构,并在此基础上开发了一个类似 Logo 语言的交互式绘图应用。

关键词 Python 脚本语言 混合语言编程

PYTHON BASED MIXED-LANGUAGE PROGRAMMING AND ITS IMPLEMENTATION

Luo Xiao Ren Yong Shan Xiuming
(*Department of Electronic Engineering, Tsinghua University, Beijing 100084*)

Abstract As a widely used script language, Python is mostly suitable for the mixed-language programming software development. This paper analyzes the features of Python and the software structure of applications developed by combining Python with system programming language. A Logo-like interactive plot application is developed to explain the implementation steps.

Keywords Python Script language Mixed-language programming

1 引 言

混合语言编程是指采用两种或两种以上的编程语言来进行程序开发的方法,一种常见情况是系统编程语言(例如 C,C++)和脚本语言(如 Tcl[1],Perl[2])之间的混合。混合语言编程的软件开发方法近年来日益受到重视[3]。采用此方法的好处有:

1) 充分利用不同编程语言的各自优势。系统编程语言开发的程序运行速度快,但开发周期较长;脚本语言方便灵活,代码简短,开发效率可提高 5-10 倍,但程序运行速度慢。合理地结合两者的优点,取长补短,可以快速高效地构建应用程序并维持相当的性能。

2) 继承历史遗留代码。虽然新的高级编程语言不断出现,但有大量重要的应用程序仍然是用 Fortran 或者 C 等老语言写成。要充分利用新语言的进展,同时使这些遗留代码继续发挥作用,把他们全部翻译成新语言是不现实的,这可以用混合语言编程的方法来加强程序的结构化和改善用户界面。

Python[4]是最近日益流行的一种脚本语言,它所具有的多种特性使其非常适用于混合语言编程。本文介绍了 Python 的特点和运用 Python 进行混合语言编程的具体方法,并基于 Python 扩展开发了一个类似 Logo 语言环境的实用交互式绘图应用。

2 Python 语言简介

Python 是一种解释型、面向对象、动态语义、语法优美的脚本语言,自从 1989 年由 Guido Van Rossum 设计出来后,经过十余年的发展,已经同 Tcl、Perl 一起,成为目前应用最广的三种跨平台脚本语言。Python 支持现有的各种主流操作系统,如 Microsoft Windows、Solaris、Mac OS、Linux 等,甚至包括 Palm OS 这样的嵌入式环境。它的源程序和二进制代码可以免费获得。由于其强大灵活的功能,简洁优美的语法和源代码免费开放,Python 被著名国际自由软件项目 KDE 计划选定为标准系统脚本语言,微软公司也宣布将在 .NET 环境中提供对 Python 语言的支持。

与同为脚本语言的 Tcl、Perl 相比,Python 的特点有:

1) *面向对象* Python 提供类,类的继承,类的私有和公有属性,例外处理等完善的对面向对象方法的支持。

2) *虚拟机* 像 Java 一样,Python 程序在执行前要先编译成字节码,再通过一个虚拟机解释执行。

3) *高级数据结构* Python 内置了对列表,关联数组等常用数据结构的支持。

4) *语法简洁优美* Python 的语法非常简单易学,并且采用缩进来表示程序的块层次结构。这样做不仅仅减少了不必要的块符号,更重要的是强制程序员用一种清晰统一的风格书写程序,增加了程序的可读性,降低了维护开销。

5) *易于扩展和嵌入* Python 语言本身只提供了一个编程语言所需功能的最小内核,其它许多丰富的功能都由扩展模块实现。由于在设计时就考虑到了扩展性,可以很方便地用 C 或者 C++ 编写 Python 的扩展模块以添加新的功能,或者把 Python 解释器自身嵌入到其他程序内部。

由于具有以上特点,Python 特别适用于混合语言编程开发。

3 基于 Python 的混合语言编程

3.1 混合语言编程软件结构框图

在混合语言编程中使用 Python 可以分为两种不同的情况:

收稿日期:2003-08-21。罗霄,硕士生,主研领域:第三代移动通信,通信系统仿真技术。

扩展和嵌入。扩展是指用 C 或 C++ 等系统语言实现 Python 的扩展模块，然后从 Python 中调用这些模块的功能。嵌入是指将 Python 解释器嵌入到应用程序中，使应用程序可以解释执行 Python 语言写成的脚本程序。扩展和嵌入都是通过 Python 的 C 语言应用程序编程接口(C API)来进行的，由此形成的软件结构如图 1 所示。C 和 Python 之间的交互主要是数据格式的转换和例外(出错)的处理。

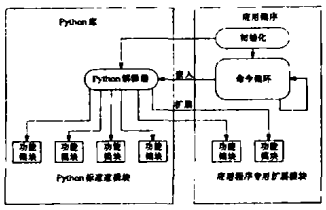

图 1 基于 Python 的混合语言编程的软件结构

3.2 扩展和嵌入的典型程序结构

一个典型的 Python 扩展模块的 C 语言源程序包括：

1) 包含 Python.h 头文件，这个文件里定义了所有 Python 的内部数据结构和 C API 函数原型；

2) 定义一个此扩展模块的例外对象；

3) 此模块定义的函数的具体实现。由函数自己负责把作为参数传递进来的 Python 对象转换为 C 程序所需的格式，处理完后再将结果转换为 Python 对象传回给 Python 解释器。如果中间出现错误，则设置此模块的例外对象并返回 NULL；

4) 填写一个此扩展模块定义的所有函数名与具体函数实现对应关系的列表；

5) 此扩展模块的初始化函数。

除了手工编写这些代码以外，还可以利用一些自动化的工具，例如 SWIG[5]。它可以扫描已有的 C 或C++ 代码，自动生成所需的 Python 扩展模块的 C 代码，这极大地提高了转换遗留代码的效率。目前，SWIG 已成功地运用于多个大型科学计算的项目开发中。

相对较为模式化的扩展来说，把 Python 嵌入应用程序的方式随着嵌入的目的不同而有很多种，而且嵌入往往也要涉及到应用程序专用的扩展，所以嵌入比扩展要稍微复杂一些，不过原理是相同的，都是通过 Python 提供的 C API 作数据格式转化和例外处理。

4 应用举例

下面以在 X 窗口环境下实现类似 Logo 语言的实用交互式绘图环境为例，介绍基于 Python 的混合语言编程。该程序实现了 reset, pendown, penup, turn, move 这几个基本的 Logo 语言命令。我们将直接利用 Python 作为语句解释器，而把绘图部分做成 Python 的扩展模块，把 Logo 语言基本命令做成扩展模块的函数。程序运行环境为：操作系统——Red Hat Linux 7.2；编译器——GCC 2.96；图形桌面环境——GNOME 1.2；Python 版本——Python 1.5.2 版。

4.1 自动生成程序框架

Python 的发行包中提供了一个工具程序 modulator，可以自动生成编写扩展模块所需的 C 语言源程序框架，我们只需在此基础上加入具体的实现代码。运行 modulator，在 Module 对话框里填上模块的名字 tortoise，然后在 Add method 对话框里加入以下几个函数名：reset, pendown, penup, turn, move。最后点击 Generate code 按钮生成 C 代码。生成的文件名最好叫作 tortoisemodule.c 以符合 Python 扩展模块的命名习惯。退出 modulator，可以看到在当前目录下多了一个叫 tortoisemodule.c 的文件。

4.2 填写具体实现代码

在 tortoisemodule.c 文件上直接进行以下修改：

1) 包含进绘图所需的 X 窗口函数头文件和宏定义；

2) 逐个修改扩展模块定义的函数，实现具体的绘图操作；以 turn 命令对应的 tortoise_turn 函数为例，修改后的函数为：

```
static PyObject * tortoise_turn(self, args)
PyObject * self;
PyObject * args;
{
/* 定义一个整形变量用来存放真正传进来的 degrees 参数 */
int degrees;
/* 从 python 对象里提取出一个整型参数赋给 degrees */
if(! PyArg_ParseTuple(args,"i",&degrees))
return NULL;
/* 真正的功能代码 */
currentDirection += (double)degrees;
/* 没有错误发生，正常返回 */
Py_INCREF(Py_None);
return Py_None;
}
```

3) 将初始化 X 和工作窗口的代码加进 inittortoise 函数。

4.3 编译执行

编写 makefile 文件，编译 tortoisemodule.c 文件，链接所需的 X 窗口库和 Python 库，生成一个叫 tortoisemodule.so 的动态链接库文件，即 Python 扩展模块 tortoise。运行 Python，包含进 tortoise 模块，执行如下 Python 语句：

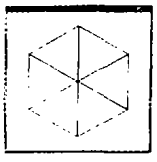

图 2 运行结果

```
>>> from tortoise import *
>>> reset()
>>> for i in range(6):
...     for j in range(3):
...         move(100)
...         turn(120)
...         turn(60)
...
>>>
```

上面的 Python 语句在一个空白窗口里画出了一个由六个等边三角形构成的六边形(如图 2 所示)，实现了类似 Logo 语言的交互式绘图功能。

在此基础上，加入系统仿真和数据分析模块，就可以容易地构成一个完整的用 Python 作为控制语言的仿真软件，例如美国洛斯阿拉莫斯国家实验室理论物理分部开发的 SPaSM 大型并行分子动力学仿真程序[6]。

5 结 语

随着计算机编程语言的发展，混合语言编程已成为一种流行的软件开发方法，Python 的特点使其非常适合于混合语言编程的软件开发。通过 Python 扩展或者嵌入，程序员可以充分利用脚本语言和系统编程语言两者的优点，达到提高开发效率，增强程序的灵活性和交互性的目的。

参 考 文 献

[1] John Ousterhout, Tcl and the Tk Toolkit[M]. Addison - Wesley, 1994.

(下转第 112 页)

医生和系统管理员。而普通操作员按照其使用 HIS 的不同部分又可以分为挂号室操作员、划价收费处操作员、医技科室操作员、病区医嘱操作员和住院管理处操作员。HIS 的角色划分如图 1 所示。

4 建立 HIS 角色用例

确定了 HIS 当中的角色，就需要通过寻找各个角色的用例确定它与 HIS 之间的关系，即角色如何通过 HIS 完成其工作或得到相应服务。在 UML 中，用例代表一个完整的功能。UML 中的用例是动作步骤的集合。动作是系统的一次执行。与角色通信，或进行计算，或在系统内工作都可以称之为动作。通过用例的分析可以确定 HIS 需要完成的功能。

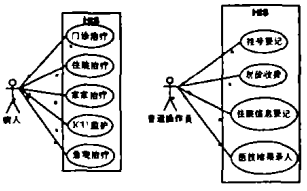

图 2 角色用例

病人可以通过 HIS 进行门诊治疗、住院治疗、家床治疗、ICU 监护和急观治疗。而系统的操作员通过使用 HIS 完成挂号登记、划价收费、住院信息登记和医技结果录入。详见图 2。

5 活动图描述角色用例

找到各个角色用例，并不能描述角色与系统交互的具体流程。这时，我们引入了 UML 的活动图。活动图用来显示动作及其结果。它着重描述操作(方法)实现中所完成的工作以及用例实例或对象中的活动，以及对象状态改革的结果。通过活动图细化各个用例，就可以了解 HIS 需要完成的工作应该怎么做。

本文以表示病人的门诊治疗过程为例，说明我们是如何在 HIS 的需求分析过程用活动图描述 HIS 的工作流程。图中，第一个泳道代表病人，第二个泳道代表挂号室(挂号室中的操作员)，第三个泳道代表信息系统(HIS)。在整个挂号过程中他们三者的协作关系如图所示。病人挂号时先检查医院中是否存在自己的病案信息，如果不存在，挂号室的操作员就帮病人新建一份病案信息，然后进入下一步；如果存在则直接进入下一步。确保 HIS 中存在病人的病案信息之后，通过病人挂号单上填写的挂号信息，挂号科室的操作员就可以通过 HIS 帮助病人建立一份挂号信息。整个挂号过程完成，如图 3 所示。

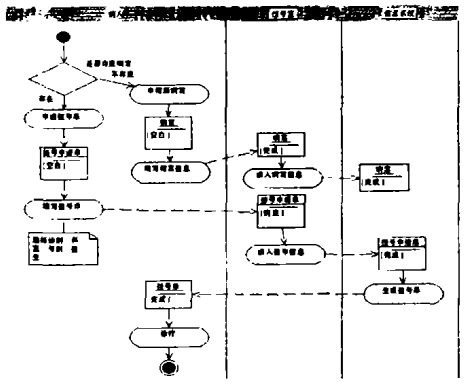

图 3 病人挂号活动图

建立了各个用例的活动图，就可以清楚的了解系统角色如何使用系统，从而完成系统的需求定义。

6 结束语

本文阐述了基于 UML 的 HIS 需求分析过程，通过形象具体的 HIS 需求分析为 HIS 的系统分析和系统实现奠定了良好的基础。

参 考 文 献

[1] Kalle Lyytinen. Different Perspectives on Information Systems: Problems and Solutions. ACM Computing Surveys, Vol. 19, No. 1, March 1987.

[2] 冯冲、王翠茹，“统一建模语言 UML 的 MIS 应用”，《中国电力》，2000, Vol.33, No.8.

[3] 蒋慧、吴礼发、称卫卫，UML Programming Guide，北京希望电子出版社，2001,1.

(上接第 18 页)

[2] Larry Wall, Tom Christiansen. Programming Perl, Second Edition [M]. O'Reilly and Associates, 1996.

[3] John Ousterhout. Scripting: Higher Level Programming for the 21st Century [J]. IEEE Computer Magazine, 1998, 31(3): 23 ~ 30.

[4] Mark Lutz. Programming Python[M], O'Reilly and Associates, 1996.

[5] David Beazley. SWIG Reference Manual[EB/OL]. http://www.swig.org.

[6] Peter Lomdahl. SpaSM's Homepage [EB/OL]. http://bifrost.lanl.gov/MD/MD.html.

(上接第 45 页)

好用的原则。电子商务平台应用模式为：用户通过网格浏览器访问网格应用服务器，进行信息交流、资金支付和安全认证等电子商务活动，其所有处理都是透明的。如图 4 示。

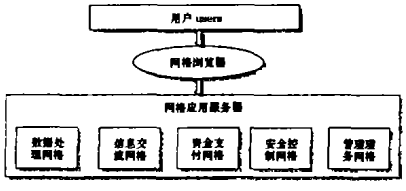

图 4

网格应用服务器是向网格最终用户提供特定服务的程序，网格应用服务器类似于现在的 Web 服务器，所不同的是 Web 服务器提供的是页面访问服务，而网格应用服务器提供的是资源访问服务。网格应用服务器通过网格编程接口实现对单个计算资源的访问或协同使用多个计算资源，通过网格服务请求协议向网格浏览器提供资源访问服务。

网格浏览器是图形化的网格客户端访问设备，实现友好的资源使用环境。网格浏览器使用网格服务标记语言，并通过协议向网格应用服务器发送资源访问请求，在计算结束后，网格应用服务器将计算结果返回给浏览器。

参 考 文 献

[1] Ian Foster, Carl Kesselman, Steven Tuecke, The Anatomy of the Grid[J], http://grid.cs.tsinghua.edu.cn, 余浩译, 2003.3.

[2] Ian Foster, What is the Grid? A Three Point Checklist[J], http://grid.cs.tsinghua.edu.cn, 王秀群译, 2003.6.

[3] 徐志伟等，“织女星网格的体系结构研究[J]”，《计算机研究与发展》，2002.8.

[4] 张英朝等，“基于网格技术的电子政务平台体系结构[J]”，《计算机应用》，2002.12.