本文由haijunzeng贡献

pdf文档可能在WAP端浏览体验不佳。建议您优先选择TXT，或下载源文件到本机查看。

用 python + wxPython 编写 Genesis Scripts

曾海军 广东梅州博敏电子 HDI 事业部工程部 navyzeng@126.com

摘要：

本文主要阐述用 python 这门编程语言配合对应的 wxPython 库来编写 Windows XP 系统下的 Genesis Scripts.

关键词: 前言：

Python、

wxPython、 Genesis Script 、 开发效率、 可维护性

Genesis 是以色列 Frontline 公司开发的一款线路板方面的计算机辅助制造软件，相对而言易学易用、 功能强大，尤其是分析和自动优化能力表现出色，该软件的优点在此不多赘述。象其他 PCB CAM 软件一 样，Genesis 也可通过编写 Scripts 来提高工作效率和减少出错。目前来看，常用来编写 Genesis Script 的工 具有：C shell+Genesis 界面命令、Perl/Tk、Tcl/Tk、Python+Python 界面库。他们都可以很好的运行在 Linux/UNIX 和 Windows 下，这四种工具各自特点如下表：

脚本语言 特点 1、C Shell 配合 sed、awk、grep 等工具，外加 Genesis 自带界面命令，足够用； 2、相对而言，勿需额外配置运行环境, 学习时间少，普及率最多，几乎可``拿来就用``; C Shell 3、Genesis 系统开发和维护不可能不涉及 C Shell，因此掌握 C Shell 也是不可缺少的； 4、许多功能实现起来相对没其他语言容易，中文支持欠缺； 5、在 Windows 操作系统下运行速度相对其他语言稍有欠缺; 6、可维护性方面相对来说最差,尤其是在开发大型 PCB CAM 应用时; 1、Perl 最大的特点就是灵活，但这是双刃剑;强大的文本处理和数据析取能力 2、须按需配置运行环境； Perl/Tk 3、相对而言，应用悠久，应用库最多，学习资料最多； 4、函数式编程和面向对象编程使代码重用率大大提高,但面向对象方面没 Python 好； 5、学习或上手时间相对更长，这也与 Perl 太灵活有很大关系； 6、因为 Perl 的太灵活，导致其可维护性方面比较差 1、Tcl 是一种可嵌入的命令脚本化语言, 小巧，易学，高效，跨平台执行; 2、须按需配置运行环境，还有 Visual Tcl 等可视化集成开发环境; Tcl/Tk 3、功能比 C shell 强大，但还是比不上 perl 和 python; 4、Tk 是基于 Tcl 的图形程序开发工具箱, 是 Tcl 的重要扩展部分，被移植到 Perl 和 Python 里 5、国内使用不是很广泛，学习资料比较少; 6、可维护性一般; 1、语法简洁、强大的内置数据结构、定位于全面开发的语言，可适用用于各种领域； 2、须按需配置运行环境; Python 3、模块化编程和面向对象编程使代码重用率大大提高,当然开发周期也大大缩短; 4、学习或上手时间相对较长，虽然易学，但要完全掌握面向对象编程有一定难度； 5、应用库没 Perl 多，相关学习资料相对比较少，近几年国内才兴起； 6、可维护性最好

笔者个人观点：兴趣是最好的老师，另须外加持之以恒；对于 Genesis Script 编写，C Shell 一定要会，

上升到一定程度，关键要选择适合自己的工具。比如原来用过 VB 语言的，用面向对象的 Python 估计会如 鱼得水，当然以前没接触过面向对象开发的，可能就来得缓点，这中间也因学习个体的素质而异，所以上 表的论断只是相对的。 本文对 Genesis 相关内部命令和变量等内容不予叙述，请参考相关文档。主要阐述用 python+wxPython 来编写 Windows 操作系统下的 Genesis Scripts，先讲述 Python 这门语言和一个优秀的界面工具包 —wxPython，再讲述具体的运行环境如何设置，然后结合一个比较典型的实例来讲解，最后是总结分析。 一、 Python 基础： Python 是一种简单易学，功能强大的编程语言，它有高效率的高层数据结构，简单而有效地实现面向 对象编程。Python 简洁的语法和对动态输入的支持，再加上解释性语言的本质，使得它在大多数平台上的 许多领域都是一个理想的脚本语言，特别适用于快速的应用程序开发。 1、 代码的缩进层次： 物理行是你在编写程序时所看见的。逻辑行是 Python 看见的单个语句。Python 假定每个物理行对应一 个逻辑行。在逻辑行首的空白（空格和制表符）用来决定逻辑行的缩进层次，从而用来决定语句的分 组,这里推荐以四个空格为基准。 2、 内置数据类型： Python 的内置数据类型包括数字、字符串、列表、字典、元组和文件。其中列表就是数学里集合的建 模，字典就是对应数学里映射的，单就这一点也说明 Python 比其他脚本语言简洁又强大。这些内置的 数据结构，也是经过优化的，运行效率非常高。一般情况下，内置数据类型就足够我们用。 运算符优先级如下表：

3、 基本语句: 从具体的角度看，Python 可以分成模块、语句和对象。他们的关系如下：程序是由模块组成的；模块 中包含语句；语句生成并处理对象；另外，Python 不象其他语言提供 switch(或 case)控制语句，只提供 以下三种控制语句： ① if 条件语句格式:

if 条件 1 : #如果条件 1 满足则执行语句 1 并忽略后续语句；否则跳过语句 1 继续

语句 1

elif 条件 2: #如果条件 2 满足则执行语句 2 并忽略后续语句；否则跳过语句 2 继续

语句 2

else: #当条件 1 和条件 2 都不满足的情况才执行语句 3

语句 3

② while 循环语句格式:

if 条件 : #如果条件满足则一直执行语句 1 并忽略后续语句；否则跳过语句 1 继续

语句 1

else: #可选的 else

语句 2

#当条件不满足时执行语句 2

③ for 循环语句格式:

for 目标 in 对象: #把对象里的每一项赋值给目标 #重复循环体（语句 1） ，使用目标

语句 1

4、 函数、模块、类: ① 函数： 象在其他语言里一样,函数是结构化编程的工具，它主要作用是代码重用和过程分解。其格式如下：

def 函数名字 (参数 1，参数 2…参数 n): 语句 return 值 #参数个数是可选的 #函数体 #返回值，当不要返回值时，可省

② 模块： 模块是包含 Python 代码的文件,为 Python 最高级的程序组织单位。我们可以通过 import 语句来导入模 块，进而使用模块文件里的代码（比如函数或类等） 。因此，也可以达到代码重用功能，但最重要的还 是模块的组件共享功能。其中环境变量 PYTHONPATH 决定了模块的查找路径。 假如在名为 m.py(模块文件应该用.py 作文件后缀名)的模块文件里包含函数 copylayer， 那么以下语句就 是先导入模块 m，再调用函数 copylayer:

import m m.copylayer() #导入模块 m #调用模块 m 里的函数 copylayer

③ 类： 类是定义一些新东西的方式， 它把现实中的对象转换到你的程序中， 它是 Python 面向对象编程 （OOP） 的主要工具。跟函数和模块一样，类是另一种打包逻辑和数据的方式，但它具有三个关键特征：多个 实例；通过继承来定制；操作符重载。类的一般形式如下：

class 类名(超类 1,超类 2,……): data = value def method(self,……): self.member.value #类用 class 语句定义，括号内为被继承的类名列表（逗号分隔，可选） #共享的类数据 #定义类里的方法 #实例的数据

5、 Python 标准库: Python 自带几百个扩展库模块，并且在不断调整和增加，包括文本字符串处理、网络和 Web 工具、数 据库接口、算法和数据结构、用户界面、数字计算等等。有了这些库，我们就能直接利用，从而减少

开发时间，使编程就象堆积木那样简单。 二、 wxPython 基础： Python 自带的图形用户界面工具是 Tkinter，它是从历史悠久的 Tcl/Tk 移植过来的。当然你可以用 Python 和自带的 Tkinter 模块来开发 Genesis 应用，那样就省得安装另外的 GUI 工具包了。但是这里我们 还是推荐更优秀的 wxPython。 wxPython 是 Python 编程语言的一个 GUI 工具箱。他使得 Python 程序员能够轻松的创建具有健壮、 功能强大的图形用户界面的程序。它是 Python 语言对流行的 wxWidgets 跨平台 GUI 工具库的绑定。任何 人都可以免费地使用它并且可以查看和修改它的源代码，或者贡献补丁，增加功能。 wxPython 程序的基本开发步骤为： ①导入必须的 wxPython 包 ②子类化 wxPython 应用程序类 ③定义一个应用程序的初始化方法 ④创建一个应用程序类的实例 ⑤进入这个应用程序的主事件循环 下面的例子很好的说明了以上步骤：

import wx class App(wx.App): def OnInit(self): frame = wx.Frame(parent=None, title='Bare') frame.Show() return True app = App() app.MainLoop() 以上代码的运行结果如下图： # 步骤 4 # 步骤 5 # 步骤 1 # 步骤 2 # 步骤 3

wxPython 模块包里含有各种可用组件, 大致又可分以下几大类，如下图：

三、 运行环境配置： 1、 Python 安装和配置： ①从 http://www.python.org/下载对应操作系统的 python 安装包，虽然有最新版的，但因为 wxPython 还 没针对最新版本 python 更新，我们这里只选 python 2.61 for windows; ②运行下载的扩展名为.msi 的安装文件，直接点``下一步``即可,跟安装其他 windows 程序类似; ③确保环境变量 PATH 值包含 Python 的安装路径和 Python 模块库的路径，保证能搜索到 Python 库; ④在环境变量 PATHEXT 的值里添加进``;.py;.pyc``，让 Windows 能直接执行该类文件； 2、 wxPython 安装：

注意要与 Python 版本和操作系统对应的; ①从 http://wxpython.org/下载 wxPython 安装包，

比如与上对应的 wxPython2.8-win32-unicode-2.8.10.1-py26.exe。 ②运行下载的安装文件，直接点``下一步``即可,跟安装其他 windows 程序类似; 3、 Python for Genesis 支持包的安装和配置： ①从 http://sourceforge.net/projects/genesis-py/下载 g2kpy\_141a.gz 压缩包; ②把 g2kpy\_141a.gz 里的 scr\_start.csh 覆盖\genesis 安装目录\eXX\all 下的同名文件; ③把 g2kpy\_141a.gz 里的以下文件放到 Python 的模块库目录下: genBasic.py、genClasses.py、genCommands.py、genFeatures.py、genPersist.py ④ g2kpy\_141a.gz 里 doc 目录下的文件为支持包的说明性文件 g2kpy\_141a.gz 里 example 目录下的文件为作者写的几个范例 四、 实例讲解：

1、 程序框架菜单 menu.py 代码举例：

#!/usr/bin/env python # -\*- coding: utf-8 -\*import os,sys,string,time,re,math import wx import genClasses import genPredo #解释性脚本语言用来标明解释器的典型写法 #标明采用 utf-8 代码 #导入 python 六个常用内置模块 #导入我们安装的 wx 模块，以便直接使用 #导入我们安装的 python for genesis 支持包，以便直接使用相关功能 #导入下面我们自己定义的模块

sys.path.append('d:/genesis/sys/scripts/python') top = genClasses.Top()

#添加搜索路径 #创建 genClasses 模块里的 Top 类的实例,注意是全局的

class MyFrame(wx.Frame): def \_\_init\_\_(self, parent, id, title):

#子类化菜单框架 #初始化菜单框架

wx.Frame.\_\_init\_\_(self, parent, id, title, wx.DefaultPosition, wx.Size(852, 42), wx.STAY\_ON\_TOP ) predo = wx.Menu() other = wx.Menu() #创建菜单类的实例 predo #创建菜单类的实例 other

#########################################加前置作业的下拉菜单############## item101 = wx.MenuItem(predo, 101, u'导入并打开料号') item101.SetFont(font) predo.AppendItem(item101) #####################################加其他制作的下拉菜单################ item1399 = wx.MenuItem(other, 1399, u'退出软件') item1399.SetFont(font) other.AppendItem(item1399) #########################添加一级菜单:前置作业和其他制作#####################

menubar = wx.MenuBar() menubar.Append(predo, u'前置作业') menubar.Append(other, u'其他制作') self.SetMenuBar(menubar) self.Centre() ##########################事件绑定##################################### self.Bind(wx.EVT\_MENU,genPredo.predo\_import, id=101) #把`导入并打开料号`菜单命令跟 genPredo 模块文件里的 predo\_import 函数挂钩 self.Bind(wx.EVT\_MENU, self.OnQuit, id=1399) def OnQuit(self,event): top.closejob() self.Close() class MyApp(wx.App): def OnInit(self): #子类化应用程序类 #初始化子应用程序类的子类 MyApp,下面为初始化内容 #把`退出软件菜`单命令跟 OnQuit 函数挂钩

#定义 OnQuit 函数 #调用 genClasses 模块里的 Top 类的 closejob 方法,关闭所有料号 #退出程序

Myframe = MyFrame(None, -1, u'梅州博敏 HDI-CAM 辅助软件 V1.0.0 ') Myframe.SetPosition(wx.Point(700,8)) Myframe.Show(True) return True

if \_\_name\_\_ == "\_\_main\_\_": app = MyApp(0) app.MainLoop() 以上代码运行结果如下： #创建一个应用程序类 MyApp 的实例为 app #进入 app 这个应用程序的主事件循环

单击导入并打开料号功能弹出类似如下对话框，选则料号后程序即自动导入并打开：

2、 模块文件 genPredo.py 举例：

下面为上面用到的模块文件 genPredo.py 里的代码内容：

#!/usr/bin/env python # -\*- coding: utf-8 -\*-

import os,sys,string,time,re,math import wx import genClasses def predo\_import(event): '''导入 ODB++形式的 Genesis 资料,并打开料号 ''' wildcard = "Genesis Job (\*.tgz)|\*.tgz|" "All files (\*.\*)|\*.\*" #定义 predo\_import 函数 #函数的说明性文档 #把过滤文件名的表达式赋值给变量 wildcard #创建打开文件对话框的实例为 dialog

dialog = wx.FileDialog(None, u'请勿选错料号', 'u:/', '', wildcard, wx.OPEN) if dialog.ShowModal() == wx.ID\_OK: jobname1 = dialog.GetFilename() jobname = jobname1[0:-4] dirpath = string.lower(dialog.GetPath()) importpath = string.replace(dirpath,'\\','/',) top.closejob() top.importJob(jobname,dirpath,'genesis') top.open\_imjob(jobname) dialog.Destroy()

#当选中 tgz 文件并点击``打开``按钮时执行以下语句 #把选中的文件名赋值给变量 jobname1 #去掉文件扩展名.tgz,并赋值给变量 jobname #自动取得文件的路径并赋值给变量 dirpath #把路径名中的`\`替换为`/`, #关闭并释放所有 Genesis 料号 #导入选中的料号到 Genesis 库 #打开刚导入的料号 #释放文件对话框的实例 dialog

注:以上代码中用到 genClasses 模块文件里的 Top 类的许多方法,这些(函数)方法的代码在 Python for genesis 支持包的相关文 件内。

五、 总结： 由上述代码可见，代码缩进风格简洁明了；在定义好了功能模块后，每次都只要简单一行代码实 现该功能，并能在各处使用；Python 编程大部分是思路的体现，很少涉及细节，是锻炼编程思路的好 语言。细节部分都大部分被写成模块里的类、类的方法或函数等，从而使代码重用率非常高。通俗的 说，就象套公式，这样就大大加快了开发速度，可读性和可维护性都非常好。 另外，Python 通过扩展模块还可操纵 ERP 数据库，还能象 VBA 一样直接操纵 Office 办公文档。 因此，Python 非常值得 PCB 行业公司用来作为全面系统维护的工具。

参考文献：

1、 《Genesis2000 自动化程序培训教材》 2、 《Python 语言入门》 3、 《wxPython in Action》 Noel Rappin & Robin Dunn 著 奥宝公司培训教材 Mark Lutz & David Ascber 著 陈革 冯大辉译 中国电力出版社