**[将Python脚本打包成可执行文件](http://www.cnblogs.com/mywolrd/p/4756005.html)**

Python是一个脚本语言，被解释器解释执行。它的发布方式：

* .py文件：对于开源项目或者源码没那么重要的，直接提供源码，需要使用者自行安装Python并且安装依赖的各种库。（Python官方的各种安装包就是这样做的）
* .pyc文件：有些公司或个人因为机密或者各种原因，不愿意源码被运行者看到，可以使用pyc文件发布，pyc文件是Python解释器可以识别的二进制码，故发布后也是跨平台的，需要使用者安装相应版本的Python和依赖库。
* 可执行文件：对于非码农用户或者一些小白用户，你让他装个Python同时还要折腾一堆依赖库，那简直是个灾难。对于此类用户，最简单的方式就是提供一个可执行文件，只需要把用法告诉Ta即可。比较麻烦的是需要针对不同平台需要打包不同的可执行文件（Windows,Linux,Mac,...）。

本文主要就是介绍最后一种方式，.py和.pyc都比较简单，Python本身就可以搞定。将Python脚本打包成可执行文件有多种方式，本文重点介绍PyInstaller，其它仅作比较和参考。

**Freezing Your Code**

各种打包工具的对比如下(来自文章[Freezing Your Code](http://docs.python-guide.org/en/latest/shipping/freezing/))：

| **Solution** | **Windows** | **Linux** | **OS X** | **Python 3** | **License** | **One-file mode** | **Zipfile import** | **Eggs** | **pkg\_resources support** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bbFreeze | yes | yes | yes | no | MIT | no | yes | yes | yes |
| py2exe | yes | no | no | yes | MIT | yes | yes | no | no |
| pyInstaller | yes | yes | yes | no | GPL | yes | no | yes | no |
| cx\_Freeze | yes | yes | yes | yes | PSF | no | yes | yes | no |
| py2app | no | no | yes | yes | MIT | no | yes | yes | yes |

PS.其中pyInstaller和cx\_Freeze都是不错的，stackoverflow上也有人建议用cx\_Freeze，说是更便捷些。pkg\_resources新版的pyInstaller貌似是支持的。

**安装PyInstaller**

对于那些网络比较稳定，能够流畅使用pip源地址的用户，直接下面的命令就可以搞定：

pip install pyinstaller

通常我们会下载源码包，然后进入包目录，执行下面的命令（需要安装setuptools）：

python setup.py install

安装完后，检查安装成功与否：

pyinstaller --version

安装成功后，就可以使用下面的命令了：

* pyinstaller : 打包可执行文件的主要命令，详细用法下面会介绍。
* pyi-archive\_viewer : 查看可执行包里面的文件列表。
* pyi-bindepend : 查看可执行文件依赖的动态库（.so或.dll文件）
* pyi-... : 等等。

**使用PyInstaller**

pyinstaller的语法：

pyinstaller [options] script [script ...] | specfile

最简单的用法，在和myscript.py同目录下执行命令：

pyinstaller mycript.py

然后会看到新增加了两个目录build和dist，dist下面的文件就是可以发布的可执行文件，对于上面的命令你会发现dist目录下面有一堆文件，各种都动态库文件和myscrip可执行文件。有时这样感觉比较麻烦，需要打包dist下面的所有东西才能发布，万一丢掉一个动态库就无法运行了，好在pyInstaller支持单文件模式，只需要执行：

pyinstaller -F mycript.py

你会发现dist下面只有一个可执行文件，这个单文件就可以发布了，可以运行在你正在使用的操作系统类似的系统的下面。

当然，pyinstaller还有各种选项，有通用选项，如-d选项用于debug，了解pyInstaller执行的过程；还有一些针对不同平台的选项，具体用法可以访问[PyInstaller官方WIKI](http://pythonhosted.org/PyInstaller)。

在执行pyInstaller命令的时候，会在和脚本相同目录下，生成一个.spec文件，该文件会告诉pyinstaller如何处理你的所有脚本，同时包含了命令选项。一般我们不用去理会这个文件，若需要打包数据文件，或者给打包的二进制增加一些Python的运行时选项时...一些高级打包选项时，需要手动编辑.spec文件。可以使用：

pyi-makespec options script [script ...]

创建一个.spec文件，对于手动编辑的.spec文件，我们可以使用下面任意一条命令：

pyinstaller specfile

pyi-build specfile

**PyInstaller的原理简介**

PyInstaller其实就是把python解析器和你自己的脚本打包成一个可执行的文件，和编译成真正的机器码完全是两回事，所以千万不要指望成打包成一个可执行文件会提高运行效率，相反可能会降低运行效率，好处就是在运行者的机器上不用安装python和你的脚本依赖的库。在Linux操作系统下，它主要用的binutil工具包里面的ldd和objdump命令。

PyInstaller输入你指定的的脚本，首先分析脚本所依赖的其他脚本，然后去查找，复制，把所有相关的脚本收集起来，包括Python解析器，然后把这些文件放在一个目录下，或者打包进一个可执行文件里面。

可以直接发布输出的整个文件夹里面的文件，或者生成的可执行文件。你只需要告诉用户，你的应用App是自我包含的，不需要安装其他包，或某个版本的Python，就可以直接运行了。

需要注意的是，PyInstaller打包的执行文件，只能在和打包机器系统同样的环境下。也就是说，不具备可移植性，若需要在不同系统上运行，就必须针对该平台进行打包。