pyinstaller 打包内置文件，ico文件内置

2019-05-23 16:52:00 weixin_30444105 阅读数 14
原文链接：http://www.cnblogs.com/darkspr/p/10912931.html

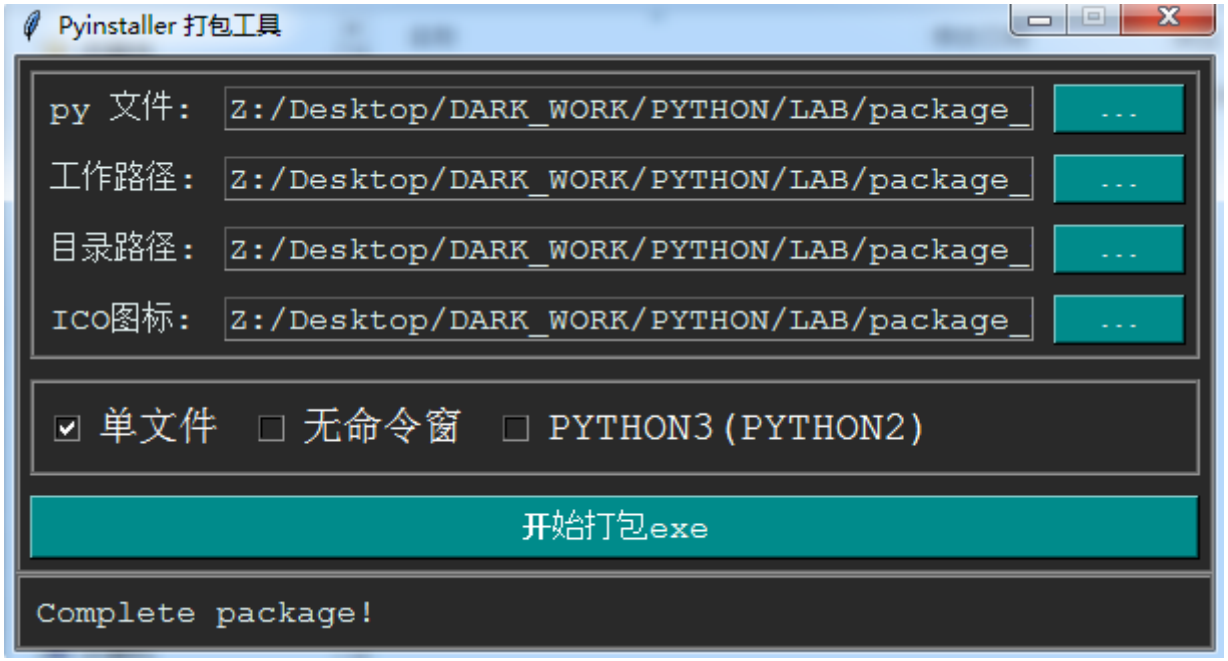
说起来用pyinstaller打包也好几年了，自从用了界面工具后，就再也没怎么深入研究过

一直以来，分发的exe都顺带一个ico文件，不然用tk写的打开就是默认logo，丑的很

然后有时候一些配置文件也要顺带发送过去，这些文件永远也不会做修改，

所以最近研究了下如何将这些资源文件打包进入exe，另外在运行的时候如何调用

首先用gui工具打包一个exe



会生成一个配置文件，并且打包exe

接下来就是看下这个配置文件了

```
# -*- mode: python -*-

block_cipher = None


a = Analysis(['Z:/Desktop/DARK_WORK/PYTHON/LAB/package_test/click.py'],
             pathex=['Z:\\Desktop\\DARK_WORK\\PYTHON\\LAB'],
             binaries=[],
             datas=[],
             hiddenimports=[],
             hookspath=[],
             runtime_hooks=[],
             excludes=[],
             win_no_prefer_redirects=False,
             win_private_assemblies=False,
             cipher=block_cipher,
             noarchive=False)
pyz = PYZ(a.pure, a.zipped_data,
             cipher=block_cipher)
exe = EXE(pyz,
          a.scripts,
          a.binaries,
          a.zipfiles,
          a.datas,
          [], 重点是这一行，资源文件都在这里
          name='click',
          debug=False,
          bootloader_ignore_signals=False,
          strip=False,
          upx=True,
          runtime_tmpdir=None,
          console=True , icon='Z:\\Desktop\\DARK_WORK\\PYTHON\\LAB\\package_test\\ico.ico')
```

如上就是配置文件，修改如上位置，加入资源文件

```
...
exe = EXE(pyz,
          a.scripts,
          a.binaries,
          a.zipfiles,
          a.datas,
          [ ('001.txt', 'Z:/Desktop/DARK_WORK/PYTHON/LAB/package_test/001.txt', 'BINARY'),
            ('ico.ico', 'Z:/Desktop/DARK_WORK/PYTHON/LAB/package_test/001.txt', 'BINARY')
          ],
          name='click',
          debug=False,
...
```

- EXTENSION：python的扩展库
- PYSOURCE：python脚本
- PYMODULE：A pure Python module (including __init__modules).
- PYZ：A .pyz archive (archive_rt.ZlibArchive)
- PKG：A pkg archive (carchive4.CArchive)
- BINARY：动态库
- DATA：数据文件
- OPTION：A runtime runtime option (frozen into theexecutable).

如上修改配置文件后进行打包 pyinstaller xxx.spec

那么如何在脚本里面调用资源呢

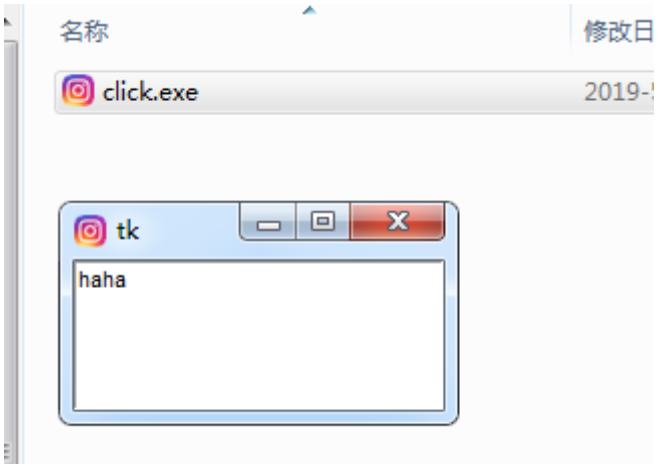
```
#!/usr/bin/env python
# coding:utf-8
import os
import sys

if getattr(sys, 'frozen', None):
    basedir = sys._MEIPASS
else:
    basedir = os.path.dirname(__file__)

data = open(os.path.join(basedir, '001.txt')).read()

from tkinter import *
top = Tk()
if os.path.exists(os.path.join(basedir, "ico.ico")):
    top.iconbitmap(os.path.join(basedir, "ico.ico"))

row1 = Frame(top)
row1.pack(fill="x")
msg = Text(row1, font=('Helvetica', '8'), width=30, height=5)
msg.pack(fill="x")
msg.insert(END, data)

# 进入消息循环
top.mainloop()
```

如上，可以看到，脚本判断自身是否是打包的exe

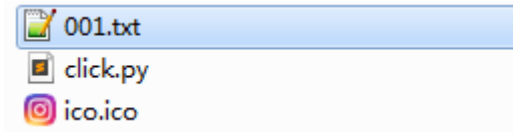
如果是的话，路径则是sys._MEIPASS,这里路径不太清楚，去temp下面找了下，没找到相关的txt和ico文件

如果不是的话，路径就是本地路径，这样开发的时候也不影响

如上打包，打开exe



打包文件则是如下，资源文件001.txt内容是haha



这样就完成了资源文件的打包和调用，总算可以发布资源的时候将logo和想要带的文件都打包进exe了

接下来有一个任务正在执行，那就是pyinstaller 加密打包

目前打包的exe，用pyinstxtractor.py即可完成破解，还原源代码，其实打包的exe，你加上什么验证，什么混淆，都是木大

都给你还原源代码，就是混淆辣眼睛一点，多花点时间也能搞定

不过我准备做的是防破解，思路如下

主要用cython，没啥特别的，将脚本编译成pyd，就行了，pyd是二进制，破解难度相当大

就是将脚本都编译成pyd

比如项目结构如下

```
utils
--tool.py           #工具集
--auth.py          #脚本验证
dark.py            #主脚本-入口
dark_api.py      #接口脚本
dark_ui.py        #UI界面脚本

dark.py
from utils.tool import *
from utils.auth import auth

from dark_api import api
from dark_ui import ui

@auth
def main():
    new_api = api()
    new_ui = ui()
    #逻辑代码
    ....
    ....
    ....
    #完成

if __name__ == '__main__':
    main()
```

那么，使用加密后

目录如下

```
utils
--tool.py           #工具集
```

```
--tool.pyd            #工具集
--auth.py             #脚本验证
--auth.pyd            #脚本验证
dark.py               #主脚本
dark.pyd              #主脚本
dark_api.py           #接口脚本
dark_api.pyd          #接口脚本
dark_ui.py            #UI界面脚本
dark_ui.pyd           #UI界面脚本
dark_run.py           #脚本入口
```

```
dark_run.py
from dark import main

main()
```

将所以的脚本都编译成了pyd，然后打包成exe

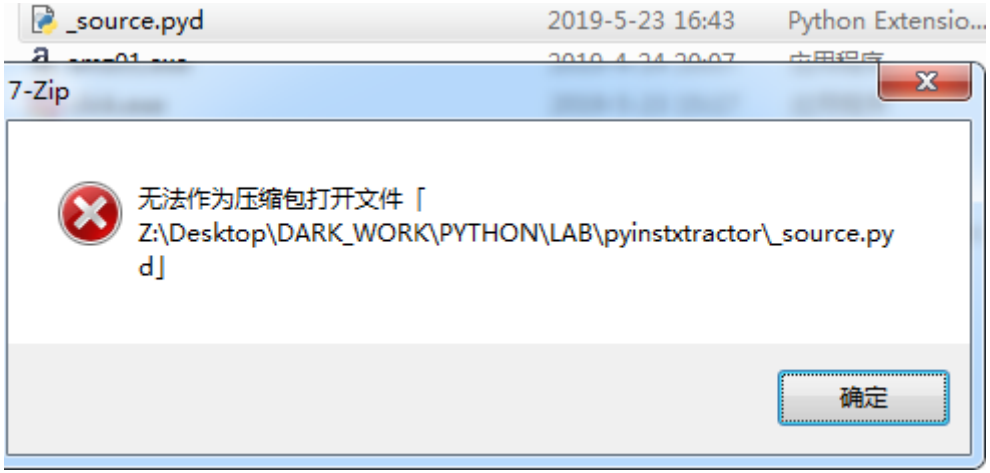
这样即使被反编译，也只能拿到pyd，拿不到pyc，脚本入口dark_run.py 可以被反编译出来，不过就算编译出来也没没用，里面就2行代码

这样就完成了代码加密保护，然后我准备再加一点料，时间长了，有可能代码丢失了，或者怎么的各种原因找不到源码了，那如果没加密，客户提供exe

还能反编译出来，自己再修改修改，现在这么一搞，就没可能了，所以我准备把源代码作为资源文件一起打包到exe里面,这是比较危险的，但也是方便自己

将源码文件打包成7z并加密，然后修改文件名为_source.pyd(伪装自己是一个pyd）在文件头部加上pyd的文件头，防止被猜出来是压缩包直接打开,这样的话

除非能想到去查看文件头，不过7z的头已经去掉，能看得出来并去掉pyd的头，加上7z的头，那就不是一般人了,至此，完成了脚本加密，源码加密隐藏内置



转载于:https://www.cnblogs.com/darkspr/p/10912931.html

文章最后发布于: 2019年05月23日 16:52:00