创建启动加载器(BUILDING THE BOOTLOADER)

PyInstaller会在有些平台，在bootloader目录下生成一个预编译的bootloader。如果没有已有的bootloader，那么pip安装程序就会尝试构建一个新的。

如果在你得平台下没有现成的预编译bootloader，或者你想修改bootloader源，你需要这么做：

使用cd 命令进入分发目录（distribution folder）

使用cd 命令进入bootloader文件夹

使用后面的命令创建bootloader python ./waf distclean all

然后使用下面的命令生成可执行的bootloader文件：

./PyInstaller/bootloader/YOUR\_OS/run,

./PyInstaller/bootloader/YOUR\_OS/run\_d

./PyInstaller/bootloader/YOUR\_OS/runw and

./PyInstaller/bootloader/YOUR\_OS/runw\_d

注意：如果你的平台上有多个版本的Python，那么你运行waf的python的配置一定是被使用过的。(python the one whose configuration is used)

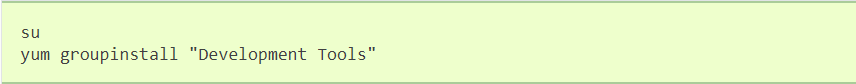
如果以上的运行过程报错，那么请仔细阅读细节的错误提示，然后据此寻求技术支持。

开发工具(Development tools)

在Debian或Ubuntu系统上，你可以运行下面的命令来安装所需的所有东西：



对于Fedora和RHEL以及它们的衍生版本，可以运行：



在Mac OS X上，你可以通过安装Xcode来获得gcc编译器。Xcode是OS X系统上的开发套件。

在Solaris 和AIX上，bootloader可以被gcc创建与测试。

在Windows上构建（Building for Windows）

在Windows平台上，你可以使用Visual Studio C++ compiler（推荐2008以上的版本）。当然免费的Visual Studio Express也可以帮你完成这件工作。

注意：bootloader是一个对使用的python没有严格版本限制的自包含静态可执行文件的东西。所以你可以使用任何你自己觉得好用的Visual Studio版本。

如果你不想使用Visual Studio这样重量的工具，也可以选择安装MinGW分发版本。

搜索以下关键词进行安装：

MinGW-64 以及 4.4以上版本的gcc

TDM-GCC-MinGW 和 MinGW-w64安装器

另外，如果你要是用MinGW-w64，则要把PATH\_TO-MINGW\bin添加到你的系统路径。也就是在构建bootloader之前运行下面的命令：



然后改变到bootloader的次级目录中，运行：



这样就会在.\PyInstaller\bootloader\YOUR\_OS目录下产生bootloader可执行文件 run\*.exe

为LINUX构建（Building for LINUX）

默认情况下，Linux系统上的bootloader是LSB二进制文件。

LSB是一系列为了增强Linux系统间的兼容性的开放标准。而PyInstaller以LSB二进制的方式生成bootloader也是为了提高最后打包的应用在linux各个分发版本系统下的稳定性和兼容性。

注意：以上需要LSB4.0以上的版本。

下面是如何在各个系统上安装LSB的内容，这里不再进行翻译。

注意：如果你不想使用LSB兼容，可以在执行命令时附加上—no-lsb参数，比如说：



与上面相同，这样就会创建support/loader/YOUR\_OS/run, support/loader/YOUR\_OS/run\_d, support/loader/YOUR\_OS/runw and support/loader/YOUR\_OS/runw\_d 这些内容，但是他们都不会是LSB二进制文件。