*摘要: pyenv 是一个类 Unix 环境下的 Python 版本管理工具。本文安装操作以ubuntu系统为例。*

# 简介

提到 Python 环境管理，更多人可能会首先想到 virtualenv。但与用于创建独立包环境的 virtualenv 不同，pyenv 的作用仅限于维护不同版本的 Python。它的使用不依赖于 Python，是一个简单、独立的纯 shell 脚本工具。pyenv 也以 [pyenv-virtualenv](https://github.com/yyuu/pyenv-virtualenv) 插件的形式支持 virtualenv，强烈建议使用。安装后比较方便的是可以在你 cd 到项目目录时自动切换相应的虚拟环境，而不用老是 source.

### 需求

出于种种原因，很多人的电脑上会同时安装很多个版本的 Python，比如会有 2.7 + 3.4。一般在 windows 下我们都可以通过使用绝对路径的方式来绕过系统对 PATH 环境变量的查询；在 Linux 下除了这种方式外，还可以在脚本文件开头显示指定需要使用的解释器，就像这样：#!/usr/bin/env python2.7 或 #!usr/bin/env python3.4。一旦特定版本的解释器被打开后，就不用再担心 PATH 的问题了。

就是有点麻烦。所以为了能让用户随心所欲地使用不同版本的诸如 python、pip、django-admin.py 这样的命令，本篇的主题，pyenv 就被开发出来了。pyenv 在安装和配置完毕后可以实现：一键（命令）切换全局、本地或当前 shell 使用的 Python 版本。

### 原理

pyenv 的美好之处在于，他并没有使用将不同的 $PATH 植入不同的 shell 这种高耦合的工作方式，而是简单地在$PATH 的最前面插入了一个垫片路径（shims）：~/.pyenv/shims:/usr/local/bin:/usr/bin:/bin。所有对 Python 可执行文件的查找都会首先被这个 shims 路径截获，从而架空了后面的系统路径。

# 安装

[pyenv 的 github 页面](https://github.com/yyuu/pyenv) 提供了完整的安装与使用指导，所以本文基本上就是对 Readme 的翻译和解释。

### 软件依赖

Linux 下安装 pyenv 前需要做一些准备工作，详情参考 [Requirements](https://github.com/yyuu/pyenv/wiki/Common-build-problems#requirements) 页面。例如 UP 使用的 Ubuntu 系统就需要先执行如下命令：sudo apt-get install -y make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm

### 自动安装

作者很亲切地提供了一个自动安装工具：

$ curl -L https://raw.githubusercontent.com/yyuu/pyenv-installer/master/bin/pyenv-installer | bash

如果想自己手动安装并进行配置的话，可参考下面的步骤。

### Github 检出

1. 将 pyenv 检出到你想安装的目录。建议路径为：$HOME/.pyenv
2. $ cd
3. $ git clone git://github.com/yyuu/pyenv.git .pyenv
4. 添加环境变量。PYENV\_ROOT 指向 pyenv 检出的根目录，并向 $PATH 添加 $PYENV\_ROOT/bin 以提供访问 pyenv这条命令的路径
5. $ echo 'export PYENV\_ROOT="$HOME/.pyenv"' >> ~/.bash\_profile
6. $ echo 'export PATH="$PYENV\_ROOT/bin:$PATH"' >> ~/.bash\_profile

这里的 shell 配置文件（~/.bash\_profile）依不同 Linux 而需作修改——Zsh：~/.zshenv；Ubuntu：~/.bashrc

1. 向 shell 添加 pyenv init 以启用 shims 和命令补完功能
2. $ echo 'eval "$(pyenv init -)"' >> ~/.bash\_profile

配置文件的位置同上一条一样需要修改

1. 重启 shell（因为修改了 $PATH）
2. $ exec $SHELL

# 使用

pyenv 总共也就 11 条不同的命令，详情可参见：[Command Reference](https://github.com/yyuu/pyenv/blob/master/COMMANDS.md#command-reference)。下面就最重要的几（8）条进行说明：

### pyenv versions

查看当前 pyenv 可检测到的所有版本，处于激活状态的版本前以 \* 标示。

$ pyenv versions

2.5.6

2.6.8

\*2.7.6 (set by /home/yyuu/.pyenv/version)

3.3.3

jython-2.5.3

pypy-2.2.1

### pyenv version

查看当前处于激活状态的版本，括号中内容表示这个版本是由哪条途径激活的（global、local、shell）

$ pyenv version

2.7.6 (set by /home/yyuu/.pyenv/version)

### pyenv install

使用 python-build（一个插件） 安装一个 Python 版本，到 $PYENV\_ROOT/versions 路径下。

$ pyenv install -v 2.7.3

建议添加 -v 参数用于显示细节。python-build 会首先尝试从一个镜像站点下载包，此时可以去 /tmp/python-build.xxx 里面关心一下下载速度。如果太慢，可以直接在 shell 里 ctrl-c 终止此次下载，然后 python-build 会自动去 python.org/ftp 下载。不一定哪个更快。

### pyenv uninstall

卸载一个版本

$ pyenv uninstall 2.7.3

### pyenv rehash

为所有已安装的可执行文件 （如：~/.pyenv/versions/\*/bin/\*） 创建 shims，因此，每当你增删了 Python 版本或带有可执行文件的包（如 pip）以后，都应该执行一次本命令

$ pyenv install 2.7.3

$ pyenv rehash

### pyenv global

设置全局的 Python 版本，通过将版本号写入 ~/.pyenv/version 文件的方式。

$ pyenv global 3.4.0

### pyenv local

设置面向程序的本地版本，通过将版本号写入当前目录下的 .python-version 文件的方式。通过这种方式设置的 Python 版本优先级较 global 高。pyenv 会从当前目录开始向上逐级查找 .python-version 文件，直到根目录为止。若找不到，就用 global 版本。

$ pyenv local 2.7.3

### pyenv shell

设置面向 shell 的 Python 版本，通过设置当前 shell 的 PYENV\_VERSION 环境变量的方式。这个版本的优先级比 local 和 global 都要高。--unset 参数可以用于取消当前 shell 设定的版本。

$ pyenv shell pypy-2.2.1

$ pyenv shell --unset

原文出处：

<https://my.oschina.net/lionets/blog/267469>