【译】使用 GNU stow 管理你的点文件

译注

这篇是翻译自 Brandon Invergo 的博客的英文文章 Using GNU Stow to manage your dotfiles 。 Brandon Invergo 的博客采用 CC-BY-SA 3.0 授权，因此本文也同样采用 CC-BY-SA 3.0 ，不同于其它我写的文章是 CC-BY-NC-SA 4.0 授权。

我自己已经使用此文中介绍的方案管理 我自己的 dotfiles 快 3 年了。最早想采用这样的管理方案是为了方便在多台 Arch Linux 系统之间同步配置， 后来逐渐主力系统也更新换代了一次，又同步到了自己的 vps 上去，目前管理多个 Arch Linux 上都多少都有这套配置。甚至装好 Arch Linux 添加好用户最初做的事情就是安装 stow git 然后 clone 了我自己的 dotfiles repo 下来，然后按需取想要的配置，快捷方便有效。

废话不多说，下面是原文和翻译。与之前的翻译一样，正文部分给出原文引用以便对照参考。

使用 GNU stow 管理你的点文件

我昨天偶然间发现一些我觉得值得分享的经验，就是那种「为毛我没有早点知道这个？」那一类的。我将在这篇文章中介绍如

I accidentally stumbled upon something yesterday that I felt like sharing, which fell squarely into the "why

何使用 GNU Stow 管理你的 GNU/Linux 系统中位于用户家目录里的各种配置文件 （通常又叫「点文件 (dotfiles)」比如 .bashrc）。

这件事的困难之处在于，如果能用版本管理系统 (VCS, Version Control System)比如 Git, Mercurial(hg), Bazaar(bzr) 管理点文件的话会非常方便，但是这些点文件大部分都位于家目录的顶级目录下， 在这个位置不太适合初始化一个版本管理仓库。这些年下来我试过很多程序，设计目的在于解决这个问题，帮你把这些配置文件安置在某个下级目录中，然后安装或者链接这些文件到它们应该在的位置。 尝试下来这些程序没有一个真正能打动我。它们要么有

the hell didn't I know about this before?" category. In this post, I'll describe how to manage the various configuration files in your GNU/Linux home directory (aka "dotfiles" like .bashrc) using GNU Stow.

The difficulty is that it would be helpful to manage one's configuration files with a version control system like Git, Mercurial or Bazaar, but many/most dotfiles reside at the top-level of your home directory, where it wouldn't be a good idea to initialize a VCS repository. Over time I've come across various programs which aim to manage this for you by keeping all the files in a subdirectory and then installing or linking them

很多依赖（比如 Ruby 和一大坨库），要么需要我记住如何用它，考虑到同步配置这种不算经常使用的场合，要记住用法真的挺难。

into their appropriate places. None of those programs ever really appealed to me. They would require a ton of dependencies (like Ruby and a ton of libraries for it) or they would require me to remember how to use them, which is difficult when really for such a task you rarely use the program.

最近我在用 GNU Stow 来管理我从源代码在本地编译安装到 `/usr/local/` 中的一些程序。 基本上说，在这种常见用法下，是你把这些本地编译的包配置安装到 `/usr/local/stow/${PKGNAME}-{PKGVERSION}` 这样的位置，然后在 `/usr/local/stow/` 目录中执行 `# stow ${PKGNAME}-${PKGVERSION}` ，然后它就会为程序所有的文件创建符号链接放在 `/usr/`

Lately I've been using GNU Stow to manage programs I install from source to /usr/local/. Basically, in this typical usage, you install locally built packages to /usr/local/stow/${PKGNAME}-{PKGVERSION} and then from /usr/local/stow/ you run # stow ${PKGNAME}-${PKGVERSION} and the program generates symbolic links to all the

`local` 中合适的地方。然后当你想用 Stow 卸载这个程序的时候，就不必再考虑会留下什么垃圾文件，或者找不到安装时用的 Makefile 了。这种安装方式下也可以非常容易地切换一个程序的不同版本（比如我想尝试不同配置选项下的 dwm 或者 st 的时候）。

programs' files into the appropriate places under /usr/local/. Then, when you uninstall a program via Stow, you don't have to worry about any stray files that you or a provide Makefile may have missed. It also makes handling alternate versions of a program quite easy (i.e. when I'm experimenting with different configurations of dwm or st).

前段时间在我扫邮件列表的时候，看到某个帖子中某人在说使用 Stow 管理安装他的点文件。 当时我没特别在意这个帖子，但是大概我大脑潜意识把它归档保存为今后阅读了。 昨天我想起来试试这种用法，试过后我不得不说，这比那些专门设计用来做这任务的点文件管理器要方便太多了， 虽然表面上看起来这种用法没那么显

Some time ago I happened across a mailing list posting where someone described using Stow to manage the installation of their dotfiles. I didn't pay much attention to it but my brain must have filed it away for later. Yesterday I decided to give it a try and I have to say that it is so much

而易见。

more convenient than those other dedicated dotfile-management programs, even if it wasn't an immediately obvious option.

方法很简单。我建了个 `${HOME}/dotfiles` 文件夹，然后在里面为我想管理的每个程序配置都 创建一个子文件夹。然后我把这些程序的配置从原本的家目录移动到这每一个对应的子文件夹中， 并保持它们在家目录中的文件夹结构。比如，如果某个文件原本应该位于家目录的顶层文件夹里， 那它现在应该放在这个程序名子目录的顶层文件夹。如果某个配置文件通常应该位于默认的 `${XDG_CONFIG_HOME}/${PKGNAME}` 位置 (`${HOME}/.config/${PKGNAME}`)， 那么现在它应该放在 `${HOME}/dotfiles/${PKGNAME}/`

The procedure is simple. I created the ${HOME}/dotfiles directory and then inside it I made subdirectories for all the programs whose cofigurations I wanted to manage. Inside each of those directories, I moved in all the appropriate files, maintaining the directory structure of my home directory. So, if a file normally resides at the top level of your home directory, it would go into the top level of the program's subdirectory. If a file normally goes in the default ${XDG_CONFIG_HOME}/${PKGN

`.config/${PKGNAME}` ，如此类推。然后在那个 dotfiles 文件夹里面，直接运行 `$ stow $PKGNAME` 命令， Stow 就会为你自动创建这些配置文件的符号链接到合适的位置。接下来就很容易为这个 dotfiles 目录初始化版本管理仓库，从而记录你对这些配置文件做的修改（并且这也可以极度简化在不同电脑之间 共享配置，这也是我想要这么做的主要原因）。

location (${HOME}/.config/${PKGNAME}) then it would instead go in ${HOME}/dotfiles/${PKGNAME} and so on. Finally, from the dotfiles directory, you just run $ stow $PKGNAME and Stow will symlink all the package's configuration files to the appropriate locations. It's then easy to make the dotfiles a VCS repository so you can keep track of changes you make (plus it makes it so much easier to share configurations between different computers, which was my main reason to do it).

举个例子，比如说你想管理 Bash, VIM, Uzbl 这三个程序的配置文件。Bash 会在家目录的顶层文件夹 放几个文件； VIM 通常会有在顶层文件夹的 .vimrc 文

For example, let's say you want to manage the configuration for Bash, VIM and Uzbl. Bash has a couple files in the top-level directory; VIM

件和 .vim 目录；然后 Uzbl 的配置位于 `${XDG_CONFIG_HOME}/uzbl` 以及 `${XDG_DATA_HOME}/uzbl` 。于是在迁移配置前，你的家目录的文件夹结构应该看起来像这样：

typically has your .vimrc file on the top-level and a .vim directory; and Uzbl has files in ${XDG_CONFIG_HOME}/uzbl and ${XDG_DATA_HOME}/uzbl. So, your home directory looks like this:

```
home/
    brandon/
        .config/
            uzbl/
                [...some files]
        .local/
            share/
                uzbl/
                    [...some files]
        .vim/
            [...some files]
        .bashrc
        .bash_profile
        .bash_logout
        .vimrc
```

然后迁移配置的方式是，应该建一个 dotfiles 子目录，然后像这样移动所有配置文件：

You would then create a dotfiles subdirectory and move all the files there:

```
home/
    /brandon/
        .config/
        .local/
            .share/
        dotfiles/
            bash/
                .bashrc
                .bash_profile
                .bash_logout
            uzbl/
                .config/
                    uzbl/
                        [...some files]
                .local/
                    share/
                        uzbl/
                            [...some files]
            vim/
                .vim/
                    [...some files]
                .vimrc
```

然后执行以下命令：

Then, perform the following commands:

```
$ cd ~/dotfiles
$ stow bash
$ stow uzbl
$ stow vim
```

然后，瞬间，所有你的配置文件（的符号链接）就安安稳稳地放入了它们该在的地方，无论原本这些目录结构 有多么错综复杂，这样安排之后的 dotfiles 文件夹内的目录结构立刻整理得有条有理，并且可以很容易地转换成版本控制仓库。非常有用的一点是，如果你有多台电脑，可能这些电脑并没有 安装完全一样的软件集，那么你可以手选一些你需要的软件配置来安装。在你的 dotfiles 文件夹中总是 可以找到所有的配置文件，但是如果你不需要某个程序的某份配置，那你就不对它执行 stow 命令，它就不会扰乱你的家目录。

And, voila, all your config files (well, symbolic links to them) are all in the correct place, however disorganized that might be, while the actual files are all neatly organized in your dotfiles directory, which is easily turned into a VCS repo. One handy thing is that if you use multiple computers, which may not have the same software installed on them, you can pick and choose which configurations to install when you need them. All of your dotfiles are always available in your dotfiles directory, but if you don't need the configuration for one

program, you simply don't Stow it and thus it does not clutter your home directory.

嗯，以上就是整个用法介绍。希望能有别人觉得这个用法有用！我知道对我来说这个非常有帮助。

Well, that's all there is to it. Hopefully someone else out there finds this useful! I know I've found it to be a huge help.