

【譯】使用 GNU stow 管理你的點文 件

譯註

這篇是翻譯自 [Brandon Invergo](#) 的博客的英文文章 [Using GNU Stow to manage your dotfiles](#)。Brandon Invergo 的博客採用 [CC-BY-SA 3.0](#) 授權，因此本文也同樣採用 [CC-BY-SA 3.0](#)，不同於其它我寫的文章是 [CC-BY-NC-SA 4.0](#) 授權。

我自己已經使用此文中介紹的方案管理我自己的 dotfiles 快 3 年了。最早想採用這樣的管理方案是爲了方便在多臺 Arch Linux 系統之間同步配置，後來逐漸主力系統也更新換代了一次，又同步到了自己的 vps 上去，目前管理多個 Arch Linux 上都多少都有這套配置。甚至裝好 Arch Linux 添加好用戶最初做的事情就是安裝 stow git 然後 clone 了我自己的 dotfiles repo 下來，然後按需求取想要的配置，快捷方便有效。

廢話不多說，下面是原文和翻譯。與之前的翻譯一樣，正文部分給出原文引用以便對照參考。

使用 GNU stow 管理你的點文件

我昨天偶然間發現一些我覺得值得分享的經驗，就是那種「爲毛我沒有早點知道這個？」那一類的。我將在這篇

I accidentally stumbled upon something that I felt like sharing, which fell into the "why the hell didn't I know about this" category. In this post, I'll describe the various configuration files in my

文章中介紹如何使用 GNU Stow 管理你的 GNU/Linux 系統中位於用戶家目錄裏的各種配置文件（通常又叫「點文件(dotfiles)」比如 .bashrc）。

這件事的困難之處在於，如果能用版本管理系統(VCS, Version Control System)比如 Git, Mercurial(hg), Bazaar(bzr) 管理點文件的話會非常方便，但是這些點文件大部分都位於家目錄的頂級目錄下，在這個位置不太適合初始化一個版本管理倉庫。這些年下來我試過很多程序，設計目的在於解決這個問題，幫你把這些配置文件安置在某個下級目錄中，然後安裝或者鏈接這些文件到它們應該在的位置。嘗試下來這些程序沒有一個真正能打動我。它們要麼有很多依

home directory (aka "dotfiles" like GNU Stow.

The difficulty is that it would be hard to manage one's configuration files with a version control system like Git, Mercurial or Bazaar. If the dotfiles reside at the top-level of your home directory, where it wouldn't be appropriate to initialize a VCS repository. Over time, you have to manage across various programs which all have their own files for you by keeping all the files in a single directory, then installing or linking them into their appropriate places. None of those solutions really appealed to me. They would require dependencies (like Ruby and a text editor) or they would require me to remember to install them, which is difficult when real world programs you rarely use the program.

賴（比如 Ruby 和一大坨庫），要麼需要我記住如何用它，考慮到同步配置這種不算經常使用的場合，要記住用法真的挺難。

最近我在用 GNU Stow 來管理我從源代碼在本地編譯安裝到 `/usr/local/` 中的一些程序。基本上說，在這種常見用法下，是你把這些本地編譯的包配置安裝到 `/usr/local/stow/` `${PKGNAME}-${PKGVERSION}` 這樣的位置，然後在 `/usr/local/stow/` 目錄中執行 `# stow ${PKGNAME}-${PKGVERSION}`，然後它就會為程序所有的文件創建符號鏈接放在 `/usr/local` 中合適的地方。然後當你想用 Stow 卸載這個程序的時候，就不必再考慮會留下什麼垃圾文件，或者找不

Lately I've been using GNU Stow to manage programs I install from source to `/usr/local/`. Basically, in this typical usage, you build packages to `/usr/local/stow/${PKGNAME}-${PKGVERSION}` and then from `/usr/local/stow/${PKGNAME}-${PKGVERSION}` you run `# stow ${PKGNAME}-${PKGVERSION}`. The program generates symbolic links for the programs' files into the appropriate location in `/usr/local/`. Then, when you uninstall Stow, you don't have to worry about the files that you or a package provide. Makefiles also makes handling alternate versions of a program quite easy (i.e. when I'm dealing with different configurations of d

到安裝時用的 Makefile 了。這種安裝方式下也可以非常容易地切換一個程序的不同版本（比如我想嘗試不同配置選項下的 `dwm` 或者 `st` 的時候）。

前段時間在我掃郵件列表的時候，看到某個帖子中某人在說使用 Stow 管理安裝他的點文件。當時我沒特別在意這個帖子，但是大概我大腦潛意識把它歸檔保存為今後閱讀了。昨天我想起來試試這種用法，試過後我不得不說，這比那些專門設計用來做這任務的點文件管理器要方便太多了，雖然表面上看起來這種用法沒那麼顯而易見。

方法很簡單。我建了個 `${HOME}/dotfiles` 文件夾，然後在裏面為我想管理的每個程序配置都創建一個子文件夾。然後我把這些程序的配

Some time ago I happened across a posting where someone described how to manage the installation of their dotfiles. I didn't pay much attention to it but my brain kept coming back to it. Yesterday I decided to try it and I have to say that it is so much easier than those other dedicated dotfile management programs, even if it wasn't an improvement.

The procedure is simple. I created a `dotfiles` directory in my home directory and then created subdirectories for all the programs I wanted to manage. For each of those directories, I moved in all the files, maintaining the directory structure.

置從原本的家目錄移動到這每一個對應的子文件夾中，並保持它們在家目錄中的文件夾結構。比如，如果某個文件原本應該位於家目錄的頂層文件夾裏，那它現在應該放在這個程序名子目錄的頂層文件夾。如果某個配置文件通常應該位於默認的 `${XDG_CONFIG_HOME}/${PKGNAME}` 位置 (`${HOME}/.config/${PKGNAME}`)，那麼現在它應該放在 `${HOME}/dotfiles/${PKGNAME}/.config/${PKGNAME}`，如此類推。然後在那個 dotfiles 文件夾裏面，直接運行 `$ stow $PKGNAME` 命令，Stow 就會為你自動創建這些配置文件的符號鏈接到合適的位置。接下來就很容易為這個 dotfiles 目錄初始化版本管理倉庫，從而記錄你

home directory. So, if a file normally goes in the top level of your home directory, the top level of the program's subdirectory normally goes in the default `${XDG_CONFIG_HOME}/${PKGNAME} (${HOME}/.config/${PKGNAME})`, then the files should instead go in `${HOME}/dotfiles/${PKGNAME}/.config/${PKGNAME}` and so on. Finally, from the dotfiles directory, just run `$ stow $PKGNAME` and Stow will move the package's configuration files to their proper locations. It's then easy to make a git repository so you can keep track of changes to the files (plus it makes it so much easier to compare configurations between different versions of the package, which was my main reason to do it).

對這些配置文件做的修改（並且這也可以極度簡化在不同電腦之間共享配置，這也是我想要這麼做的主要原因）。

舉個例子，比如說你想管理 Bash, VIM, Uzbl 這三個程序的配置文件。Bash 會在家目錄的頂層文件夾放幾個文件；VIM 通常會有在頂層文件夾的 .vimrc 文件和 .vim 目錄；然後 Uzbl 的配置位於

`${XDG_CONFIG_HOME}/uzbl` 以及

`${XDG_DATA_HOME}/uzbl`。於是在遷移配置前，你的家目錄的文件夾結構應該看起來像這樣：

For example, let's say you want to have configuration for Bash, VIM and Uzbl. You have a couple files in the top-level directory; Bash has your .vimrc file on the top-level directory; and Uzbl has files in `${XDG_CONFIG_HOME}/uzbl` and `${XDG_DATA_HOME}/uzbl`. So, your directory looks like this:

```
1  home/
2      brandon/
3          .config/
4              uzbl/
5                  [...some files]
6          .local/
7              share/
8                  uzbl/
9                      [...some files]
10         .vim/
11             [...some files]
12         .bashrc
13         .bash_profile
14         .bash_logout
15         .vimrc
```

然後遷移配置的方式是，
應該建一個 dotfiles 子目
錄，然後像這樣移動所有
配置文件：

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


```
1  home/
2      /brandon/
3          .config/
4          .local/
5              .share/
6      dotfiles/
7          bash/
8              .bashrc
9              .bash_profile
10             .bash_logout
11      uzbl/
12          .config/
13              uzbl/
14                  [...some fil
es]
15          .local/
16              share/
17                  uzbl/
18                      [...some
files]
19      vim/
20          .vim/
21              [...some files]
22          .vimrc
```

然後執行以下命令：

Then, perform the following commands:

```
1 $ cd ~/.dotfiles
2 $ stow bash
3 $ stow uzbl
4 $ stow vim
```

然後，瞬間，所有你的配置文件（的符號鏈接）就安安穩穩地放入了它們該在的地方，無論原本這些目錄結構有多麼錯綜複雜，這樣安排之後的 dotfiles 文件夾內的目錄結構立刻整理得有條有理，並且可以很容易地轉換成版本控制倉庫。非常有用的一點是，如果你有多臺電腦，可能這些電腦並沒有安裝完全一樣的軟件集，那麼你可以手選一些你需要的軟件配置來安裝。在你的 dotfiles 文件夾中總是 可以找到所有的配置文件，但是如果你不需要某個程序的某份配置，那你就不對它執行 stow 命令，它就不會擾亂你的家目錄。

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

嗯，以上就是整個用法介紹。希望能有別人覺得這個用法有用！我知道對我來說這個非常有幫助。

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

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.