stow

A command-line configuration management tool

Davis Claiborne

LUG @ NC State

February 13, 2019

Linux Users Group
at NC State University

Introduction

Stow is a command-line utility program used to manage config files

Consider the following problems:

- Managing dotfiles using VCS can be difficult
- Cleaning up a program's installation can be difficult

Stow solves both of these problems in an easy to use, intuitive way

Your dotfiles without stow

Imagine you have the following config files in your home directory:

```
.
├── .bashrc
├── .config
│   ├── beets
│   │   └── config.yaml
│   ├── i3
│   │   └── config
│   ├── i3status
│   ├── ranger
│   │   └── rc.conf
│   └── zathura
│       └── zathurarc
├── .xinitrc
└── .Xresources
```

Your dotfiles with stow

Stow lets you configure your dotfiles like this:

```
dotfiles
├── bash
│   ├── .bashrc
│   └── .inputrc
├── beets
│   └── .config
│       └── beets
│           └── config.yaml
├── i3
│   └── .config
│       ├── i3
│       │   └── config
│       └── i3status
│           └── config
│
...
```

How to use / benefits

To populate these directories:

```
cd ~/dotfiles
stow bash
stow beets
stow i3
...
```

Pros:

- Easier to manage with VCS
- Logically separate config files

Cons:

- Can be confusing
- If structure of files change, symlinks can become dead [1]

[1] This can be easily handled using the -D or -R flag

Your home directory with stow

Symlinks now exist in the appropriate directories for your dotfiles

```
.
├── .bashrc -> dotfiles/bash/.bashrc
├── .config
│   ├── beets
│   │   └── config.yaml -> ~/dotfiles/beets/.config/beets/...
│   ├── i3
│   │   └── config -> ~/dotfiles/i3/.config/i3/config
│   ├── i3status -> ~/dotfiles/i3/.config/i3status
│   ├── ranger
│   │   └── rc.conf -> ~/dotfiles/ranger/.config/ranger/...
│   └── zathura
│       └── zathurarc -> ~/dotfiles/zathura/.config/...
├── .xinitrc -> dotfiles/x11/.xinitrc
└── .Xresources -> dotfiles/x11/.Xresources
```

How it works

When you run `stow <directory>`, `stow` goes through `<directory>` and finds new files and directories to symlink [2]

By default, the fewest possible symlinks are created

[2] Also checks to see if file/directory matches the ignore list. More on that later.

Additional use cases

Stow can be used to do more than just manage config files... it can be used to manage program installation itself.

For instance, `imagemagick` creates 16 files in `/usr/local/man/man1`:

- imagemagick
- convert
- compare
- composite
- identify
- ...

If you're not using a package manager, or if you're using a custom installation, you'll have to remember that all of these files will need to be removed when you uninstall it.

Managing program installs with stow

Make a directory where all the files will go, e.g. `/usr/local/stow/`

When installing a program, set it to install to that directory, either during the `configure` step or the make step, depending on how the build system works. [3]

Now you can run `stow` exactly how you did before (with proper privileges) to install the program.

[3] There can be some issues with programs when they are installed in one location but look for packages in another area. The solution to this can be seen in section 12 [3]

Ignoring files

Certain files can be ignored to prevent them from being stowed.

This can be done in the following ways:

- With the `--ignore` flag
- `~/.stowrc` file
- `~/.stow-global-ignore` to always ignore
- `.stow-local-ignore` for case-by-case situations

Perl-style regex can also be used to ignore certain files

Ignoring files

An example `~/.stow-global-ignore` file:

```
# Comments and blank lines are allowed.

\.git           # Excludes any and all .git files/directories
\.gitignore

^/README.*      # Excludes only top-level READMEs
^/LICENSE.*
^/COPYING
```

Patterns with / are matched exactly (/ indicates the top of the directory being stowed)

If the pattern does not contain a forward slash, matching is based on `basename`

Multi-environment configuration

You can also manage your config-files on a per-environment basis if you have different installation scenarios for different machines:

```
~/dotfiles/
        work/
                .config/
                .icewm/
                .mutt/
        server/
                .tmux.conf
        common/
                .vimrc
                .vim/
                        bundle/
                        colors/
                .bashrc
                .bashrc_aliases
```

Alternatives to stow

- Ansible [1]
  - Developed by Red Hat
  - Mostly used in industry setting
- rcm [2]
  - RC Manager
  - Intended to be more flexible stow (features hooks, tags, etc.)
  - Written in POSIX sh
  - Generates install script
- vcsh [4]
  - Version Control System for $HOME
  - Syntax is intended to mimic Git's
- yadm [5]
  - Yet Another Dotfile Manager
  - Also based on Git syntax
  - Great documentation

Advantages of stow

Benefits of stow (in my opinion):

- Very few dependencies [4] and easy to install
- Mature project
- Flexibile nature makes it environment agnostic
- Can be used for more than just dotfiles
- KISS

[4] Only dependency is Perl

Disadvantages of stow

Drawbacks of stow:

- Simplicity $\rightarrow$ less customization (no hooks, etc.)
- Documentation can be poor/vague
- Can make experimentation more difficult

References I

[1] Ansible https://www.ansible.com/

[2] rcm https://thoughtbot.com/blog/rcm-for-rc-files-in-dotfiles-repos

[3] Stow https://www.gnu.org/software/stow/manual/stow.html

[4] vcsh https://github.com/RichiH/vcsh

[5] yadm https://thelocehiliosan.github.io/yadm/