Using `stow` To Manage Your Dotfiles

A few weeks ago I made a post about keeping track of your dotfiles using `rsync`. While not a horrible way to do it, there is a **way** better way of handling it using `stow`. I first heard about it here on Wolfgang's Channel. It took me a second to get a real grasp for it, but it is pretty straight forward.

This post is going to talk about what stow is and how I currently use it. s

What is `stow`?

Pulling this quote directly from the GNU page:

> GNU Stow is a symlink farm manager which takes distinct packages of software and/or data located in separate directories on the filesystem, and makes them appear to be installed in the same place. For example, /usr/local/bin could contain symlinks to files within /usr/local/stow/emacs/bin, /usr/local/stow/perl/bin etc., and likewise recursively for any other subdirectories such as . . . /share, . . . /man, and so on.
>
> This is particularly useful for keeping track of system-wide and per-user installations of software built from source, but can also facilitate a more controlled approach to management of configuration files in the user's home directory, especially when coupled with version control systems.
>
> Stow is implemented as a combination of a Perl script providing a CLI interface, and a backend Perl module which does most of the work. Stow is Free Software, licensed under the GNU General Public License.

In other words, it symlinks the desired file based of a starting directory and the path dictated by the file tree created for the file. If that doesn't make sense, don't worry. Take a look at the following examples and it will make more sense.

Example 1

You have a vimrc file in `~/.vimrc`. You want to symlink that. So you have a git repository that you cloned in `~/Code/dotfiles`. If you want to keep track

of your vimrc file, what you do is first do is create a directory called vim (or whatever else you want to call it) in your dotfiles directory. After doing so, move your vimrc into the vim directory you just created. Now, if you type `stow ~/ vim` while you are in the dotfiles directory you will symlink you vimrc to `~/.vimrc`.

The `-t` flag sets that starting location for where you want the symlink to propagate from. So from the point of view from `stow` you are looking in the directory `vim` that has the path `.vimrc`. Since you are starting from `~/` you will symlink the `.vimrc` file to `~/.vimrc`.

Example 2

You have a config file `~/.config/nvim/init.vim` that you want to keep track of. Following the steps from the previous example, you create a directory called `nvim` in your dotfiles and create a file structure `nvim/.config/nvim/`. Copy the init.vim file so you have the structure `nvim/.config/nvim/init.vim'. To symlink this file you type`stow -t ~/ nvim‘.

Again, from the point of view of `stow`, you are linking the path `.config/nvim/init.vim` starting at `~/` creating the link `~/.config/nvim/init.vim`.

My `stow` Workflow

As a TL;DR, my workflow is essentially what I described above. The only difference is that I have a script that has all the name of the directories appended to the stow command. The only difference is that I also keep track of my packages in a different script by running `pacman -Qqe > PackageList.txt` which creates a list of all the packages you installed in such a way that you can use it as an input to pacman later on. So at the end of the day, you have a little scrip that looks like the following:

`setConfig.sh`

```
#!/bin/sh

stow -t ~/ aliasrc dunst git Xmodmap zshenv compton env nvim shortcutrc zsh xinitrc rofi mbs
```

`updatePackages.sh`

```
pacman -Qqe > PackageList.txt
```