GNU Stow

Symlink farm manager

START HERE

`stow <pkg>`
Link a package into the parent dir

`stow -n -v <pkg>`
Dry run, show what would link

`stow -t ~ <pkg>`
Link into home

`stow -D <pkg>`
Remove the links

`stow -R <pkg>`
Relink after changes

`stow -d <dir> <pkg>`
Use another stow dir

`stow --adopt <pkg>`
Pull existing files in

`stow --dotfiles <pkg>`
dot-bashrc becomes .bashrc

CONCEPTS

Each package is a directory that mirrors the target tree. Stow symlinks its contents, so ~/dotfiles/vim/.vimrc becomes ~/.vimrc.

Stow dir ~/dotfiles
↓ stow <pkg>
Package vim/.vimrc
↓ symlink
Target dir ~/.vimrc

The default target is the parent of the stow dir, so keep the repo in ~/dotfiles or pass -t ~.

Stow only removes symlinks it owns and reports a conflict rather than overwriting a real file.

STOW AND UNSTOW

`stow <pkg>...`
Link packages (default action)

`stow -S <pkg>`
Explicit stow action

`stow -D <pkg>`
Delete a package's links

`stow -R <pkg>`
Restow: unlink then link again
Also cleans up links to files you removed from the package.

`stow -n <pkg>`
Simulate, change nothing

`stow -v <pkg>`
Verbose output (repeat up to -vvvvv)

`stow --verbose=<n> <pkg>`
Set verbosity level 0-5

`stow -D <old> -S <new>`
Mix actions in one run

DIRECTORIES

`stow -d <dir> <pkg>`
Set the stow directory

`stow -t <dir> <pkg>`
Set the target directory

`stow -t ~ <pkg>`
Link into your home directory

`STOW_DIR=<dir> stow <pkg>`
Stow dir from environment

`stow --dir=<dir> --target=<dir> <pkg>`
Long form of -d and -t

DOTFILES AND CONFLICTS

`stow --dotfiles <pkg>`
Turn dot- prefix into a leading dot

`stow --adopt <pkg>`
Move existing target files into package
Overwrites package content with the target's version. Review with git diff and restore with git checkout.

`stow --no-folding <pkg>`
Link files, never whole directories

`stow --override=<regex> <pkg>`
Take over paths owned by another package

`stow --defer=<regex> <pkg>`
Skip paths already stowed elsewhere

--adopt rewrites the files in your stow dir. Commit first so git diff shows what it took, then git checkout . to keep your repo version.

IGNORING AND CONFIG

`stow --ignore=<regex> <pkg>`
Skip names ending in a Perl regex

`~/.stowrc`
Default options, one per line

`./.stowrc`
Per-directory default options

`.stow-local-ignore`
Per-package ignore patterns file

`~/.stow-global-ignore`
Ignore patterns for all packages

`stow -V`
Print version

`stow -h`
Show command syntax

TREE FOLDING

Folding behaviour

Situation	Result
`Target dir missing`	One symlink to the whole directory
`Target dir already real`	Links go inside it, file by file
`Second package needs folded dir`	Symlink split into real dir, both linked
`Package removed, one left`	Directory refolded to one symlink
`--no-folding`	Always file-level links

TREE FOLDING (cont.)

If an app writes new files into a folded directory they land in your repo. Use --no-folding or pre-create the directory.

RECIPES

First-time dotfiles setup

```
mkdir -p ~/dotfiles/vim
mv ~/.vimrc ~/dotfiles/vim/.vimrc
cd ~/dotfiles && stow -nv vim
stow vim
```

New machine

```
git clone <url> ~/dotfiles
cd ~/dotfiles
stow -nv bash vim git   # check conflicts
stow bash vim git
```

Adopt existing configs safely

```
cd ~/dotfiles && git status   # must be clean
stow --adopt vim
git diff                      # see what was adopted
git checkout .                # keep repo version
```

Install software under /usr/local

```
cd /usr/local/stow
sudo stow -v <pkg-1.0>
sudo stow -D <pkg-1.0>
```