Warp Directory

Version v1.7.0

Table of Contents

[Downloads] 

[FOSSA Status] | *https://app.fossa.com/api/projects/git%2Bgithub.com%2Fkigster%2Fwarp-*

dir.svg?type=large

This is a ruby implementation of the tool `wd` (warp directory), originally written as a `ZSH` module by Markus Færevaag.

I personally went back to `bash` after trying out `ZSH`, but it was the `wd` plugin that I really missed.

While Markus kindly offered a ruby version in a separate branch of this module, it wasn't quite as extensible as I wanted to (or well tested), so it ended up being an inspiration for this gem.

Chapter 1. Warp This

WarpDir is a UNIX command line tool that works somewhat similar to the standard built-in command `cd` — "change directory".

The main difference is that `wd` is able to add/remove/list folder "shortcuts", and allows you to jump to these shortcuts from anywhere on the filesystem.

This of this as a folder-navigation super-charge tool that you'd use on a most frequently-used set of folders. This becomes **really useful** if you are often finding yourself going into a small number of deeply nested folders with a long path prefix.

Chapter 2. Installation

Three steps:

- `wd` requires a Ruby interpreter version 2.2 higher.
  - Please Check your default ruby with `ruby --version`. You should see something like "ruby 2.3.0p0....".
  - If you see version 1.9 or earlier, please upgrade your ruby using the package manager native to your OS.
- Install `warp-dir` ruby gem (note: you may need to prefix the command with `sudo` if you are installing into the "system" ruby namespace).

```
$ gem install warp-dir
```

- The last step is to install the `wd` BASH function and auto-completion. This step appends the required shell function to your shell initialization file, that is specified with the `warp-dir install --dotfile <shell-dot-file>` command:

```
$ warp-dir install --dotfile ~/.bash_profile
Shell support is installed in the following files:
/Users/kig/.bash_profile
$ source ~/.bash_profile
# Now we can use 'wd' shortcut
$ wd --help
```

After the last step you **need to restart your session**, so — if you are on Mac OS X, — please reopen your Terminal or better yet — iTerm2, and then type:

```
$ wd help
```

If the above command returns a properly formatted help that looks like the image below, your setup is now complete!

```
Usage:  wd [ --command ] [ list | help ] [ wd-flags ]
        wd [ --command ] [ [ warp ] | add  [-f] | rm | ls ] [ wd flags ] <point> -- [ cmd-flags ]
        wd --help | help

Warp Point Commands:

      add <point>     Adds the current directory as a new Warp Point, aka: new, save, store
       ls <point>     List directory contents of a Warp Point, aka: dir, ll
   remove <point>     Removes a given warp point from the database, aka: rm, delete
     warp <point>     Jumps to the pre-defined warp point (command optional)

Global Commands:

    clean             Removes any no-longer existing warp points, aka: x
  install             Installs warp-dir shell wrapper in your ~/.bashrc
     help             Show this extremely unhelpful text, aka: wtf
     list             Print all stored warp points, aka: l

Examples
       wd add proj    # add current directory as a warp point
       wd ~/          # wd works just like "cd" for regular folders
       wd proj        # jumps to proj
       wd list        # lists all "bookmarked" points
       wd rm proj     # removes proj

Installation
    It appears that you already have a wrapper installed in one of the default
    shell init files (~/.bash_profile, ~/.bashrc, ~/.profile, ~/.bash_login).

    Which means that "wd" should be working on your system.  If not, edit your
    shell init file, remove any lines related to warp-dir gem, and then reinstall:

    wd install [ --dotfile <filename> ]

    If you experience any problem, please log an issue at:
    https://github.com/kigster/warp-dir/issues

    -m, --command  <command>      - command to run, ie. add, ls, list, rm, etc.
    -p, --point    <point-name>   - name of the warp point
    -w, --warp     <warp-point>   - warp to a given point
    --no-color                    - do not use ASCII color
    -f, --force                   - force, ie. overwrite existing point when adding
    -h, --help                    - show help
    -v, --verbose                 - enable verbose mode
    -q, --quiet                   - suppress output (quiet mode)
    -d, --debug                   - show stacktrace if errors are detected
    -s, --dotfile  <dotfile>      - shell init file to append the wd wrapper, eg. ~/.bashrc
    -c, --config   <config>       - location of the configuration file (default: ~/.warprc)
    -V, --version                 - print the version
```

Chapter 3. Usage

in the below examples, the characters `~ □ ` denote the current shell prompt, showing the current folder you are in. The command to type is on the right hand side of the "□".

Let's first bookmark a long directory:

```
~ □ cd ~/workspace/arduino/robots/command-bridge/src
~/workspace/arduino/robots/command-bridge/src □ wd add cbsrc
Warp point saved!

~/workspace/arduino/robots/command-bridge/src □ cd ~/workspace/c++/foo/src
~/workspace/c++/foo/src □ wd add foosrc
Warp point saved!

~/workspace/c++/foo/src □ cd /usr/local/Cellar
/usr/local/Cellar □ wd add brew
Warp point saved!
```

Now we can list/inspect current set of warp points:

```
/usr/local/Cellar □ wd l
   cbsrc -> ~/workspace/arduino/robots/command-bridge/src
  foosrc -> ~/workspace/c++/foo/src
    brew -> /usr/local/Cellar
```

Now we can jump around these warp points, as well as run 'ls' inside (even passing arbitrary arguments to the `ls` itself):

```
/usr/local/Cellar □ wd cbsrc
~/workspace/arduino/robots/command-bridge/src □ wd foosrc
~/workspace/c++/foo/src □  1 wd ls brew -- -alF | head -4        # run ls -alF inside
/usr/local/Cellar
total 0
drwxrwx---  73 kig  staff  2482 May  7 15:29 ./
drwxrwx---  21 kig  staff   714 Apr 28 11:40 ../
drwxrwx---   3 kig  staff   102 Dec 24 03:14 ack/
```

3.1. Command Completion in BASH

If you installed `wd` properly, it should register it's own command completion for BASH and be ready for your tabs :)

Note that you can use `wd` to change directory by giving an absolute or relative directory name, just like `cd` (so not just using warp-points), so when you type `wd <TAB><TAB>` you should see the list of *all*

saved warp points as well as all of the local sub-directories relative to where you are at.

```
# And, it supports command completion in BASH!
$ wd<TAB><TAB>
# should print the list of registered warp points, and commands.

$ wd install --dotfile /Users/kig/.bash<TAB><TAB>
/Users/kig/.bash_login    /Users/kig/.bash_profile  /Users/kig/.bashrc
```

Command completion is activated by loading the `~/.bash_wd` file that's installed with `warp-dir install` command.

3.2. Config File (aka. Warp Points Database)

All of the mappings are stored in the `~/.warprc` file, where the warp point name is followed by a colon, and the path it maps to. So it's trivial to do a global search/replace on that file in your favorite editor, if, for example, a commond top level folder had changed.

The format of the file was left identical to that of the `ZSH` version of `wd` so that one could switch back and force between the two versions of `wd` and still be able to use their collection of warp points.

See? I think we thought of everything :)

Happy warping!

Chapter 4. `wd` Concept

The overall concept comes from the realization that when we work on the command line, we often do things that `wd` tool provides straight out of the box, such as:

- we often have to deal with a limited number of folders at any given time
- on occastion have to jump between these folders (which we call **warp points**), which may require mult-level `cd` command, for example: `cd ~/workspace/foo/src/include/; ....; cd ~/Documents/Microsoft\ Word/; ...`
- seems like it should be easy to add, remove and list warp points
- everything should require typing few characters as possible :)
- it would be great to have full BASH completion support

Some future extensions could be based on some additional realizations:

- perhaps you might want to inspect a bookmarked folder without leaving your current place.
- maybe by inspecting we mean — running a `find`, or `ls` or any other command for that matter

4.1. Notable Differences with original `wd`

- instead of `wd add!` use `wd add -f <point>` (or --force)

These features will be added shortly:

- for now `wd clean` is not supported
- for now history is not supported
- for now '-' is not supported

4.2. Future Feature Brainstorm

4.2.1. Simplify The CLI

Questionable value, but this sort of interface appear a bit more consistent.

Still I am not sure I want to type `wd -j proj` or `wd -a proj` instead of `wd proj` and `wd add proj`...

```
wd -j/--jump   point
wd -a/--add    point
wd -r/--remove point
wd -l/--ls     point
wd -p/--path   point

wd -L/--list
wd -C/--clean
wd -S/--scan           # report whether points exist on the file system
```

Chapter 5. Run Commands In A Warp Point

Pass an arbitrary command to execute, and return back to CWD.

```
wd proj -x/--exec -- "command"
```

5.1. Networking

Can we go across SSH?

```
wd add proj kig@remote.server.com:~/workspace/proj
wd ls proj
wd proj
```

This then establishes and SSH connection to the server and logs you into the shell. Should be pretty easy, I think :)

Chapter 6. Development

Fork the repo to your github username, and create a feature branch. Run `bundle install`.

You can also run `bin/console` for an interactive prompt that will allow you to experiment.

To submit your change, create a new pull request, and ensure to provide tests for any new code.

Chapter 7. Adding New Commands

Just follow the pattern in the `lib/warp/dir/commands/` folder, copy and modify one of the existing commands. Command class name is used as an actual command.

Add a working rspec.

Chapter 8. Contributing

Bug reports and pull requests are welcome on GitHub at https://github.com/kigster/warp-dir.

Chapter 9. Author

Chapter 10. License

This project is distributed under the MIT License.