what Documentation

Release 0.7.0

J Alan Brogan

Jan 03, 2019

Contents

CHAPTER 1

What is whyp?

whyp extends *type*, [wittily](https://www.reddit.com/r/commandline/comments/2kq8oa/the_most_productive_function_i_have_written/clo0gh2/)

whyp shows the source of aliases, functions and executables available in the shell

CHAPTER 2

What is eype?

eype edits commands whether aliases, functions or scripts

CHAPTER 3

Names

whyp devolved from a repo called *what*, which extended *which*. *what* got a bit too big for it's boots, so I yearned for the minimalistic tendencies of *type*. So *whyp* was born as a smashing of "what" and "type"

eype is a reminder that silly names are one honking great idea – let's do more of those!

Official prounciation guide *here <https://www.youtube.com/watch?v=tXo0o3dg4vQ>_*. Official video *here <https://www.youtube.com/watch?v=RidtrSCogg0>_*.

3.1 Install

Clone the repo

$ git clone https://github.com/jalanb/whyp.git

Go there

$ cd whyp

Install to your current *bash* shell

$ source __init__.sh

If you want to keep *whyp* for bash, then

$ echo "" >> ~/.bashrc $ echo "source $(readlink -f __init__.sh)" >> ~/.bashrc

Install to *python*

$ deactivate # (if required)

$ pip install -r requirements.txt $ python setup.py develop

3.2 whyp in bash

Now you can use *whyp* as a replacement for *type* in current shell

$ type ls ls is /bin/ls

$ whyp ls ls is /bin/ls

3.3 whyp in python

$ python >>> from whyp import ls >>> assert ls == '/bin/ls'

CHAPTER 4

Daily use

Simplified versions of the commands outlined below are provided for quick use These versions are designed to be simple to use, so are short and clustered near the *w* key.

w will call *whyp* to show the type of something

$ w python python is /usr/local/bin/python

ww will show more about the command (definition of the alias or conents of the function/file)

$ ww cd /usr/bin/cd #!/bin/sh # $FreeBSD: src/usr.bin/alias/generic.sh,v 1.2 2005/10/24 22:32:19 cperciva Exp $ # This file is in the public domain. builtin *echo ${0##*/} | tr [:upper:] [:lower:]* ${1+"$@"}

e will call *eype* to edit something

$ e whyp

CHAPTER 5

whyp

whyp searches within aliases, functions and files to show the sources. For example, let's imagine you set up an alias to make cd'ing to the parent directory easier, e.g.

$ alias up='cd ..'

Then the whyp command can show that up is an alias

$ whyp up alias up='cd ..'

If you add the verbose flag, -v, then it will look "into the alias" and run whyp on that command too. In this example "cd" turns out to be a file, so that is shown too

$ whyp up -v alias up='cd ..' -r-xr-xr-x 15 root wheel 190 Mar 25 18:03 /usr/bin/cd

#!/bin/sh # $FreeBSD: src/usr.bin/alias/generic.sh,v 1.2 2005/10/24 22:32:19 cperciva Exp $ # This file is in the public domain. builtin *echo ${0##*/} | tr [:upper:] [:lower:]* ${1+"$@"}

Let's re-write up as a function

$ unalias up $ up () { > cd .. > }

Then the whyp command can show that up is a function.

$ whyp up up is a function

If you add the verbose flag, -v, then the source of the function is shown

$ whyp up -v #! /bin/bash up () {

cd ..

}

The up command could also be written as a file

$ unset up $ echo "#! /bin/bash" > ~/bin/up $ echo "cd .." >> ~/bin/up $ chmod +x ~/bin/up

And the whyp command would then find the file, and show a long *ls* for it

$ whyp up -rwxr-xr-x 1 jalanb staff 19 Apr 24 22:56 /home/jalanb/bin/up

If you add the verbose flag, -v, then the source of that file is shown too

$ whyp -v up -rwxr-xr-x 1 jalanb staff 19 Apr 24 22:56 /home/jalanb/bin/up #! /bin/bash cd ..

If you add the quiet flag, -q, then no output is shown, which can be useful when using *whyp* in scripts

$ if whyp -q ls; then echo yes; else echo no; fi yes

If you add the errors flag ('-e') then the script will hide some errors shown in bash commands.

Some bash commands can show errors, but give a successful return code. With '-e' the script runs, and hides these error messages. Without '-e' the script fails and shows the error (this is default)

CHAPTER 6

whypyp

The *whypyp* command tries to find where python will import files from, for example

    $ whypyp os -rw-r–r– 1 root root 24258 Sep 19 2006 /usr/lib/python2.7/os.py

Note that the whypyp command uses whatever the default installation of python is, hence in the example above it found the module used for the python2.7 installation. The python version can also be specified as an argument to find imports for other python installations. Such a version argument must be the first argument.

    $ whypyp 2.6 os -rw-r–r– 1 root root 26300 Aug 12 2012 /usr/local/lib/python2.6/os.py

The author has used the scripts on * OSX 10.7 and 10.11 using python 2.5, 2.6 and 2.7 with bash 3.2.48 and 4.3.42 * CentOS 5.0, 6.5 and 7.0 using python 2.4 and 2.7 with bash 3.2.25, 4.1.2 and 4.2.46 * Ubuntu 10.04 and 12.04 using python 2.6 and 2.7 with bash 4 * babun 1.2.0 (on cygwin on Windows 7) using python 2.7.8 with bash 4.3.33

and continues to use them many times per day.

CHAPTER 7

Witty

One of the best compliments of my coding life was when [/u/andru183 called this code “witty”](https://www.reddit.com/r/commandline/comments/2kq8oa/the_most_productive_function_i_have_written/clo0gh2/)

CHAPTER 8

Version

0.7.0 Version: 0.7.0

Contents:

CHAPTER 9

Indices and tables

- genindex
- modindex
- search