keycut

Release 0.3.0

Mar 10, 2019

Contents

CHAPTER 1

keycut

A command line tool that helps you remembering ALL the numerous keyboard shortcuts of ALL your favorite programs.

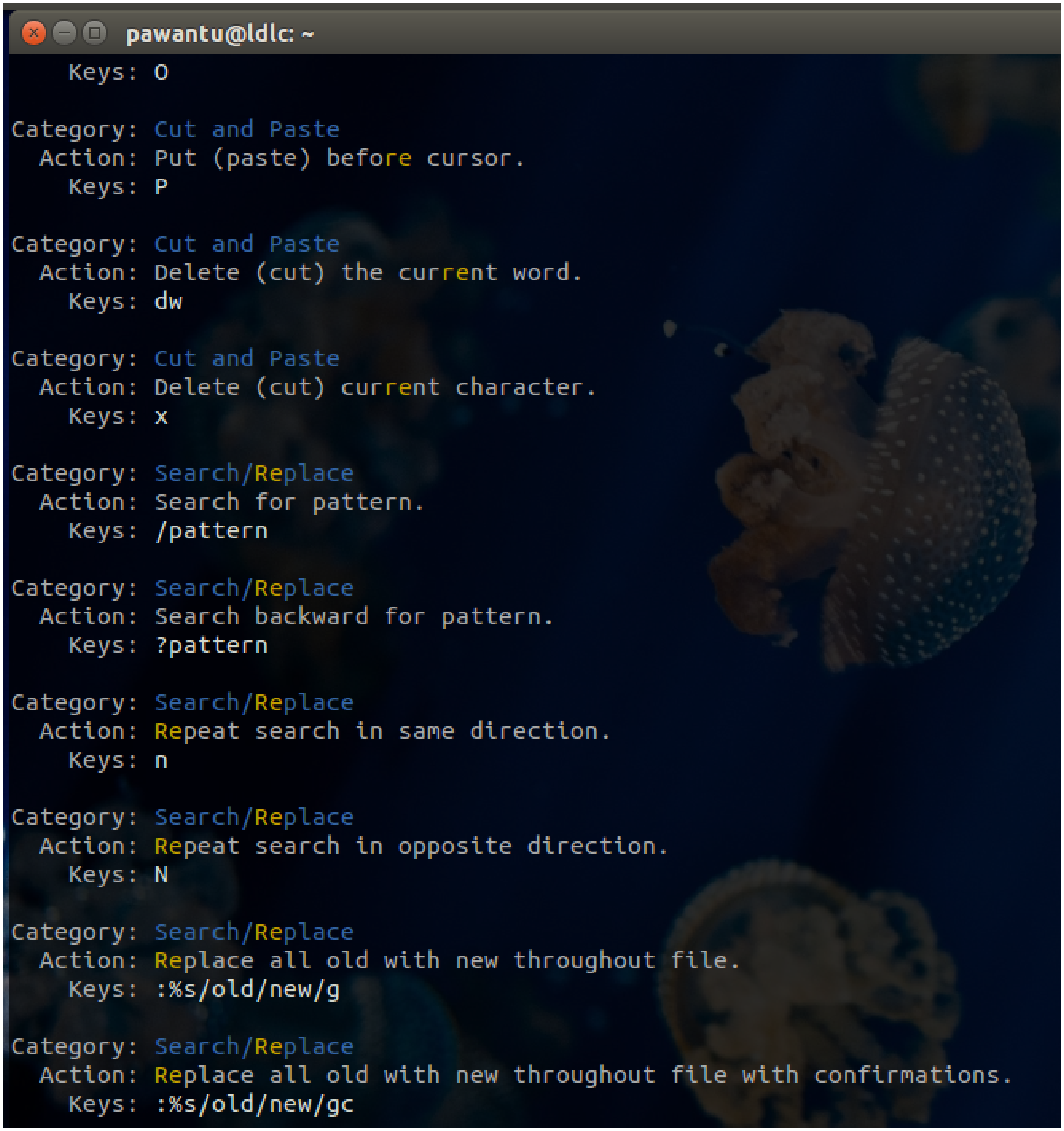
KeyCut (for keyboard shortcut) is a command line tool that helps you remembering the numerous keyboard shortcuts of your favorite programs, both graphical and command line ones, by allowing you to print them quickly in a console and search through them.

Shortcut data are provided by the [keycut-data][1].

This repository contains the sources for a Python implementation of KeyCut.

1.1 How it looks

The yellow parts are the one that matched a pattern using a regular expression.



```
    Keys: O

Category: Cut and Paste
  Action: Put (paste) before cursor.
    Keys: P

Category: Cut and Paste
  Action: Delete (cut) the current word.
    Keys: dw

Category: Cut and Paste
  Action: Delete (cut) current character.
    Keys: x

Category: Search/Replace
  Action: Search for pattern.
    Keys: /pattern

Category: Search/Replace
  Action: Search backward for pattern.
    Keys: ?pattern

Category: Search/Replace
  Action: Repeat search in same direction.
    Keys: n

Category: Search/Replace
  Action: Repeat search in opposite direction.
    Keys: N

Category: Search/Replace
  Action: Replace all old with new throughout file.
    Keys: :%s/old/new/g

Category: Search/Replace
  Action: Replace all old with new throughout file with confirmations.
    Keys: :%s/old/new/gc
```

1.2 Requirements

keycut requires Python 3.6 or above.

To install Python 3.6, I recommend using pyenv.

```
# install pyenv
git clone https://github.com/pyenv/pyenv ~/.pyenv

# setup pyenv (you should also put these three lines in .bashrc or similar)
```

(continues on next page)

(continued from previous page)

```
export PATH="${HOME}/.pyenv/bin:${PATH}"
export PYENV_ROOT="${HOME}/.pyenv"
eval "$(pyenv init -)"

# install Python 3.6
pyenv install 3.6.8

# make it available globally
pyenv global system 3.6.8
```

1.3 Installation

With pip:

```
python3.6 -m pip install keycut
```

With pipx:

```
# install pipx with the recommended method
curl https://raw.githubusercontent.com/cs01/pipx/master/get-pipx.py | python3

pipx install --python python3.6 keycut
```

You will also need to download the data by cloning the repository somewhere:

```
git clone https://github.com/pawamoy/keycut-data ~/.keycut-data
```

1.4 Usage

The program needs to know where the data are. By default, it will search in the (relative) keycut-data/default directory.

```
export KEYCUT_DATA=~/.keycut-data/default
```

Show all bash shortcuts:

```
keycut bash
```

Show all bash shortcuts matching *proc* (in Category, Action, or Keys):

```
keycut bash proc
```

Command-line help:

```
usage: keycut [-h] APP [PATTERN]

Command description.

positional arguments:
  APP         The app to print shortcuts of.
  PATTERN     A regex pattern to search for.
```

(continues on next page)

(continued from previous page)

```

optional arguments:
  -h, --help  show this help message and exit
```

CHAPTER 2

Contributing

Contributions are welcome, and they are greatly appreciated!

Every little bit helps, and credit will always be given.

For bug reports, feature requests, and feedback, simply create a new issue. Try to be as descriptive as possible.

2.1 Bug fixes, new features and documentation

This project is developed using `poetry`. Follow the recommended installation method:

```
curl -sSL https://raw.githubusercontent.com/sdispater/poetry/master/get-poetry.py |␣
↪python
```

Then follow these instructions:

1. Fork the repository on github.com;
2. Clone it on your machine;
3. Go into the directory, and run `poetry install` to setup the development environment;
4. Create a new branch with `git checkout -b bug-fix-or-feature-name`;
5. Code!
6. **Write tests. Run them all.** The commands to run the tests are:

   ```
   poetry run pytest  # to run all tests sequentially
   poetry run pytest -v  # to print one test per line
   poetry run pytest -n 4  # to run tests in parallel (4 workers)
   poetry run pytest tests/test_api.py  # to run tests in a specific file
   ```

   `pytest` provides the `-k` option to select tests based on their names:

```
poetry run pytest -k "api and remove"
poetry run pytest -k "utils or stats"
```

See the documentation for the `-k` option for more examples.

A Makefile is available for convenience: `make test`.

7. When the tests pass, commit (make sure to have atomic commits and contextual commit messages! Check out this awesome blog post by Chris Beams for more information.)

8. Push;

9. . . . and finally, create a new pull/merge request! Make sure to follow the guidelines.

CHAPTER 3

Contributor Covenant Code of Conduct

3.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

3.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

3.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

3.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

3.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at pawamoy@pm.me. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

3.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

CHAPTER 4

Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

4.1 v0.3.0 (compare) - 2019-03-10

4.1.1 Refactor

- The project now uses Poetry! (294fc17).

4.2 v0.2.1 (compare) - 2017-02-28

4.2.1 Changed

- Change license in README (06eed4b).

4.2.2 Fixed

- Fix bandit warnings and template merge changes (5e08d38).
- Fix flake8 (5ea78f1).
- Fix imports and tox configuration (e20a74e).
- Fix tests (data path), change license from MPL 2.0 to ISC (6ad4b07).

4.3 v0.2.0 (compare) - 2016-12-28

4.3.1 Added

- Add acknowledgments in README (b4727e6).

4.3.2 Fixed

- Fix `main.py` (2594310).
- Fix missing dotfiles… (dea1209).

4.4 v0.1.0 (compare) - 2016-01-15

4.4.1 Added

- Add image to README (a898097).
- Add usage in README (8abb66b).
- Added contents (e79fec5).
- Added data contents (145e50e).
- Added README (4efb0bc).
- Added source code (9c32fd2).

4.4.2 Fixed

- Fix README syntax (5ffa491).
- Fix redundancy and badly colored keys (20c55d9).

CHAPTER 5

Credits

These projects were used to build keycut. **Thank you!**

| poetry | cookie-poetry

5.1 Direct dependencies

bandit | black | flake8 | ipython | isort | jinja2-cli | pytest | pytest-cov | pytest-sugar | pytest-xdist | PyYAML | recommonmark | safety | Sphinx | sphinx-rtd-theme | sphinxcontrib-spelling | termcolor | toml

5.2 Indirect dependencies

alabaster | apipkg | appdirs | appnope | atomicwrites | attrs | Babel | backcall | certifi | chardet | Click | colorama | | coverage | decorator | docutils | dparse | entrypoints | execnet | gitdb2 | GitPython | idna | imagesize | ipython-genutils | jedi | | MarkupSafe | mccabe | more-itertools | packaging | parso | pbr | pexpect | pickleshare | pluggy | prompt-toolkit | ptyprocess | py | pycodestyle | pyenchant | pyflakes | Pygments | pyparsing | pytest-forked | pytz | requests | six | smmap2 | snowballstemmer | sphinxcontrib-websupport | stevedore | traitlets | urllib3 | wcwidth

More credits from the author

See one of your project without the logo? Make sure it's available on VectorLogoZone (GitHub repo) and open an issue or send a pull/merge request!