delete

Release 1.1.3

Apr 24, 2020

Contents:

A simple, cross-platform, command line move-to-trash.

Release v1.1.3. (*Installation*)

CHAPTER 1

But why?

The `delete` command is a simple alternative to using the standard `rm` command. Using `rm` as a matter of course can be dangerous and prone to mistakes. Once a file is unlinked with `rm` it cannot be recovered (without having backups).

All major graphical environments offer a "move to trash" option. This does a clean move operation to a "trash" folder. Once a file as been put in the trash it can be recovered easily. Periodically, the trash can be emptied if desired.

`delete` is a command line implementation of this metaphor. It maintains a basic `sqlite3` database of files and folders put in the trash. Using the `--list` option will list the contents. Using `--restore` will restore a file or folder from the trash. Using `--empty` will purge anything put in the trash by `delete`.

1.1 Getting Started

1.1.1 Installation

If you already have Python 3.7 on your system, you can install `delete` using Pip.

```
pip install delete-cli
```

1.1.2 Basic Usage

Calling `delete` with no arguments or with the `--help` flag yield typically Unix style behavior, print a usage or help statement, respectively. For detailed usage and examples you can read the manual page, `man delete`.

Deleting files and folders is as simple as:

```
delete file1.txt file2.txt folderA
```

Files or folders that get deleted with the same basename will have a suffix added before the extension (e.g., `file1.1.txt`, `file1.2.txt`, ...).

Restore files using their basename (in the trash), their full path (in the trash) or their original full path.

```
delete --restore file1.txt
delete --restore $TRASH_FOLDER/file2.txt
delete --restore /original/path/folderA
```

List the contents of the trash along with their original full paths.

```
delete --list
```

Use `--empty` to completely empty the trash. This does not remove the `$TRASH_FOLDER`. It iterates through the full listing of contents from the `$TRASH_DATABASE`. If anything is left in the directory an error is logged.

```
delete --empty
```

1.1.3 Recommendations

Add the following to your shell's login profile to shorten the invocation.

```
alias del="delete"
```

1.1.4 Known Issues

- On *macOS* systems the default `~/.Trash` is protected and does not allow a listing of the directory. `delete` functions normally aside from an error message being printed when using `--empty`.

1.2 Configuration

`delete` has no real configuration other than where on the system to keep the trash folder and its database, both of which are controlled by the below environment variables.

1.2.1 Environment Variables

TRASH_FOLDER

By default, the program will move objects to the `~/.Trash` folder. This is for consistency with systems like macOS and most Linux distributions. In this way, the *empty trash* operation on these systems will also remove anything put there by the `delete` program via the command line. You can specify a different location via the `TRASH_FOLDER` environment variable. For example, in your `~/.bashrc`.

```
export TRASH_FOLDER=$HOME/.deleted
```

TRASH_DATABASE

By default, the path to the database is `${TRASH_FOLDER}.db`, which would be `~/.Trash.db` if the `TRASH_FOLDER` environment variable is undefined. This path *should not* be inside the trash folder. You can specify a different location via the `TRASH_DATABASE` environment variable. For example, in your `~/.bashrc`.

```
export TRASH_DATABASE=$HOME/.my_trash.db
```