
cparse Documentation

Release 0.0.2

Lucian Cooper

Apr 14, 2019

Commands

1	Installation	3
2	Usage	5

Current version: 0.0.2
Github
PyPi

CHAPTER 1

Installation

Use pip via PyPi:

```
pip install cparse
```

Or clone the repository:

```
git clone git://github.com/luciancooper/cparse.git
cd cparse
python setup.py install
```


CHAPTER 2

Usage

cparse is a command line tool. There are currently 6 subcommands:

ls	list files in directory
tree	print file tree
stat	directory filetype stats
py	python code parsing
html	html link parsing
css	css code parsing

2.1 ls

The `ls` command lists the files in a directory

2.1.1 Usage

```
cparse ls [-r] [-n <depth>] [-d | -f] [-a] [-lim <count>] [-fmt <format>]
          [-exc <path>] [-inc <path>]
          [-wc <pattern>] [-grep <regex>] [-ft <filetype>]
          [-m | -M | -c | -C | -b | -B | -i | -I | -g | -G] <path>
```

2.1.2 Positional Arguments

<path>	root directory
--------	----------------

2.1.3 Optional Arguments

<code>-r</code>	list files recursively
<code>-n <depth></code>	max depth if recursive flag is specified
<code>-d</code>	dirs only flag
<code>-f</code>	files only flag
<code>-a</code>	include hidden files
<code>-lim <count></code>	maximum items to list in output
<code>-fmt <format></code>	display format for listed items

2.1.4 Sorting Flags

Control the order in which files are listed. *Only one of the following flags can be specified.*

<code>-m</code>	sort by modified time (most recent first)
<code>-M</code>	sort by modified time (least recent first)
<code>-c</code>	sort by created time (newest first)
<code>-C</code>	sort by created time (oldest first)
<code>-b</code>	sort by size (largest first)
<code>-B</code>	sort by size (smallest first)
<code>-i</code>	sort by inode (descending)
<code>-I</code>	sort by inode (ascending)
<code>-g</code>	group files by file extension (descending)
<code>-G</code>	group files by file extension (ascending)

2.1.5 Pruning Arguments

Control which sub directories to include when recursive flag is specified. *These arguments can be specified multiple times.*

<code>-exc <path></code>	sub paths to exclude
<code>-inc <path></code>	sub paths to include

2.1.6 Filtering Arguments

Apply filters to control which files are listed.

<code>-wc <pattern></code>	wild card pattern
<code>-grep <regular-expression></code>	regular expression to match
<code>-ft <file-extension></code>	file type filter

2.2 tree

The `tree` command prints file trees

2.2.1 Usage

```
cparse tree [-d | -f] [-a] [-n <depth>] [-fmt <format>]
           [-exc <path>] [-inc <path>]
           [-wc <pattern>] [-grep <regular-expression>] [-ft <file-extension>]
           [-m | -M | -c | -C | -b | -B | -i | -I | -g | -G] <path>
```

2.2.2 Positional Arguments

<path>	tree root directory
--------	---------------------

2.2.3 Optional Arguments

-d	dirs only flag
-f	files only flag (ignore empty directories)
-a	include hidden files
-n <depth>	max tree depth
-fmt <format>	display format for tree nodes

2.2.4 Sorting Flags

Control the order in which files are listed within each branch of the tree. *Only one of the following flags can be specified.*

-m	sort by modified time (most recent first)
-M	sort by modified time (least recent first)
-c	sort by created time (newest first)
-C	sort by created time (oldest first)
-b	sort by size (largest first)
-B	sort by size (smallest first)
-i	sort by inode (descending)
-I	sort by inode (ascending)
-g	group files by file extension (descending)
-G	group files by file extension (ascending)

2.2.5 Pruning Arguments:

Control which sub directories to include in tree. *These arguments can be specified multiple times.*

-exc <path>	sub paths to exclude from tree
-inc <path>	sub paths to include in tree

2.2.6 Filtering Arguments

Apply filters to control which files are included in the tree.

-wc <pattern>	wild card pattern
-grep <regular-expression>	regular expression to match
-ft <file-extension>	file type filter

2.3 stat

The `stat` command produces a table displaying directory filetype proportions

2.3.1 Usage

```
cparse stat [-a] <path>
```

2.3.2 Positional Arguments

<path>	root directory
--------	----------------

2.3.3 Optional Arguments

-a	include hidden files
----	----------------------

2.4 py

The `py` command parses python code files

2.4.1 Usage

```
cparse py <path>
```

2.4.2 Positional Arguments

<path>	either a directory to search for .py files in, or a .py file
--------	--

2.5 html

The `html` command parses the links in html files

2.5.1 Usage

```
cparse html <path>
```

2.5.2 Positional Arguments

<path>	either a directory to search for html files in, or a html file
--------	--

2.6 css

The `css` command parses css code

2.6.1 Usage

```
cparse css [-g] [-c] [-s] <path>
```

2.6.2 Positional Arguments

<path>	a css file to parse
--------	---------------------

2.6.3 Optional Arguments

-g	group identical selector property blocks
-c	condense redundancies within property blocks
-s	stack matching selectors in output