fmap Documentation

Release 0.1

Matt Bodenhamer

April 18, 2016

Contents

A Python command-line utility for recursively applying a command to a filesystem tree.

The program works by walking the filesystem tree (either the value of `-r`, if supplied, or the current working directory) using Python's `os.walk`. The program is invoked with a string specifying a command to be executed at each directory in the tree. In each directory, each file (and sub-directory, if `-d` is specified) is matched against the list of include and exclude patterns specified at the command line. If a file (and/or directory) matches an include pattern and does not match any exclude patterns, the command is executed with that file (or directory) name as an argument. If no include patterns are specified, the program applies the command to all files (or directories) that do not match any exclude pattern.

CHAPTER 1

Installation

```
$ pip install -U fmap
```

CHAPTER 2

Usage

```
Usage: fmap [-h] [-p] [-v] [-d] [-l] [-b] [-z <depth>] [-x <pattern>] [-r <dir>]
     <cmd> [<pattern> [<pattern> ...]]

Recursively apply a command to a filesystem tree.

positional arguments:
  <cmd>                 The command to apply. The file to be applied may be
                        optionally specified by '{}'. If '{}' is not
                        supplied, the file will be passed in as the last
                        argument.
  <pattern>             Unix filename pattern that specifies which files to
                        apply the command to.

optional arguments:
  -h, --help            Show this help message and exit
  -p, --preview         Doesn't apply the command. Instead, prints command
                        invocations that would be performed.
  -v, --verbose         Print command invocations as they are performed.
  -d, --apply-dirs      Apply the command to directories after it is applied
                        to files at each level of the tree.
  -l, --follow-links    Follow symbolic links.
  -b, --bottom-up       Walk the tree from the bottom up. By default, the tree
                        is traversed from the top down.
  -z <depth>, --max-depth <depth>
                        Maximum recursion depth. Any negative number results
                        in unlimited recursion. Default is -1.
  -x <pattern>, --exclude <pattern>
                        Unix pattern that specifies which files to exclude
                        applying the command to.
  -r <dir>, --root <dir> Directory in which to begin the traversal. Is the
                        current directory by default.
```

CHAPTER 3

Examples

Suppose you have a directory structure under `/root`, like so:

```
.profile
file1
file2
dir1/
    file3
    dir3/
        file4
dir2/
    file5
```

As a trivial example, to list all the file paths under the current directory:

```
$ pwd
/root

$ fmap echo
/root/file1
/root/file2
/root/.profile
/root/dir2/file5
/root/dir1/file3
/root/dir1/dir3/file4
```

To exclude certain patterns, use the `-x` option:

```
$ fmap -x .profile echo
/root/file1
/root/file2
/root/dir2/file5
/root/dir1/file3
/root/dir1/dir3/file4

$ fmap -x '.*' -x file[35] echo
/root/file1
/root/file2
/root/dir1/dir3/file4
```

Remember to quote any wildcard patterns to prevent them from being expanded by the shell.

By default, the command is applied to all files. However, you can specify include patterns after the command:

```
$ fmap echo file1 file[35]
/root/file1
/root/dir2/file5
/root/dir1/file3
```

Include and exclude patterns can be combined:

```
$ fmap -x file[35] echo 'file*'
/root/file1
/root/file2
/root/dir1/dir3/file4
```

By default, the command is not applied to directories. This can changed, however, by supplying `-d`:

```
$ fmap -d echo
/root/file1
/root/file2
/root/.profile
/root/dir2
/root/dir1
/root/dir2/file5
/root/dir1/file3
/root/dir1/dir3
/root/dir1/dir3/file4
```

The command is applied to directories after it has been applied to all applicable files at that level.

By default, the file tree is walked top-down. To walk the tree bottom-up, supply the `-b` option:

```
$ fmap -b echo
/root/dir2/file5
/root/dir1/dir3/file4
/root/dir1/file3
/root/file1
/root/file2
/root/.profile
```

To print out the command invocation as it is executed, supply `-v`:

```
$ fmap -v echo
echo /root/file1
/root/file1
echo /root/file2
/root/file2
echo /root/.profile
/root/.profile
echo /root/dir2/file5
/root/dir2/file5
echo /root/dir1/file3
/root/dir1/file3
echo /root/dir1/dir3/file4
/root/dir1/dir3/file4
```

To preview which command invocations will take place without actually invoking them, use the `-p` option:

```
$ fmap -p 'rm -f' 'file*'
rm -f /root/file1
rm -f /root/file2
rm -f /root/dir2/file5
```

```
rm -f /root/dir1/file3
rm -f /root/dir1/dir3/file4
```

However, no files will actually be deleted using the above command.

Remember to quote the command invocation if it includes arguments or subcommands. You can also use `{}` to specify where the file path should be inserted into the command invocation:

```
$ fmap -v 'echo {} >> out'
echo /root/file1 >> /root/out
echo /root/file2 >> /root/out
echo /root/.profile >> /root/out
echo /root/dir2/file5 >> /root/out
echo /root/dir1/file3 >> /root/out
echo /root/dir1/dir3/file4 >> /root/out

$ cat out
/root/file1
/root/file2
/root/.profile
/root/dir2/file5
/root/dir1/file3
/root/dir1/dir3/file4
```

CHAPTER 4

Changelog

4.1 0.1 (2016-04-16)

Initial release