Vim as the poor man's sed

briantorti.com/vim-as-the-poor-mans-sed

February 17, 2015

Not long ago I wrote about `sed`, a powerful non-interactive editor that can be used to edit multiple files in a fairly easy way. Today I want to show how we could use `vim`'s not so well known `ex` mode to do some of these same tasks, and what are the benefits and shortcomings.

The `ex` mode

The `ex` mode is very similar to the `command` mode: It allows you to enter ex commands. The main difference is that you won't be back to `normal` mode after the command is executed. You can enter the `ex` mode with `Q`, and go back to `normal` mode with `:visual`.

It's a mode designed for batch processing, and we can start `vim` with `-e`, if that's all we need.

The usage

The usage is very similar to what we did with `sed`. We just need to give it a file path and a set of commands to be executed:

```
$ echo "foo bar baz" > testing-ex.txt
$ vim -e testing-ex.txt <<-SCRIPT
%s/foo/new-value
w
SCRIPT
```

The same way, you could just pipe the result of an `echo` to vim:

```
# Note that, in vim, "|" is used to execute multiple commands at once:

$ echo "%s/foo/new-value/ | w" | vim -e testing-ex.txt
```

Or we could just move this script to its own file, and then execute the command as:

```
$ vim -e testing-ex.txt < command.vim
```

This script is just a bunch of `vim` commands, the same commands you would execute if you were editing this file by hand. The only caveat is to remember that you need to save the file (`w`) in the end.

For comparison, the equivalent `sed` command would be something like this:

```
$ sed -i'' -f command.sed testing-ex.txt
```

The benefit

The benefit is that it's just `vim`, and you probably already know the commands to edit a file. If you have a map to do some kind of editing, you are all set, just execute these commands in `ex` mode. What if you want to join all the lines? Just execute `%join`, as you would if you were editing a single file.

> Check :help ex-cmd-index for the list of all the ex commands available

The shortcomings

I know you thought about that as soon as you read the title of this post: Performance. And yes, you are right, `Vim` won't be that fast if you need to edit a lot of files. Let's measure that by running the same script that substitutes a string in 10.000 files:

```
# Creates 10.000 files to test
$ for i in {1..1000}; do echo "test" > $i.txt; done

$ time for i in {1..1000}; do echo "%s/test/new-value/g | w" | vim -e $i.txt; done
# Executes in 22.13s

$ time for i in {1..1000}; do sed -i'' -e 's/test/new-value/g' $i.txt; done
# Executes in 3.12s
```

In this simple test, `sed` is about 8x faster.

Conclusion

There is some overlap in what you can do with `sed` and with `vim` in `ex` mode. There is no right or wrong, they are just options, and, as always, it's important to know the trade offs and when it's worth to use one option over another.

Interested in learning Kubernetes?

I just published a new book called Kubernetes in Practice, you can use the discount code **blog** to get 10% off.

Get fresh articles in your inbox

If you liked this article, you might want to subscribe. If you don't like what you get, unsubscribe with one click.