How to split out pieces of a file while preserving git line history: The hard way with commit-tree

devblogs.microsoft.com/oldnewthing/20190917-00

September 17, 2019

Raymond Chen

Last time, we looked at how to split a single file into multiple files while preserving line history. A related scenario is where you want to extract some pieces of a file into separate files, but leave some pieces behind.

Let's use the same scratch repo we had last time. You can follow the same copy/paste script, or you can take your existing scratch repo and `git reset --hard ready` to get it back into its "ready to start experimenting" state.

First, we're going to do things the hard (but more information-theoretically correct) way, and then we'll develop a simpler alternative that gets the same result, though through some potentially-confusing intermediate steps.

Okay, to do things the hard way, we split out each file in its own branch.

```
git checkout -b f2f

git mv foods fruits
git commit --author="Greg <greg>" -m "create fruits from foods"
```

We start by renaming `foods` to `fruits`. This ensures that when git traces the history of the `fruits` file, it will follow the history back into the `foods` file.

Next, we split the `fruits` file back into two files: The fruits stay in the `fruits` file, and the rest go back into the `foods` file.

```
>foods echo celery
>>foods echo cheese
>>foods echo eggs
>>foods echo lettuce
>>foods echo milk
>>foods echo peas
git add foods

>fruits echo apple
>>fruits echo grape
>>fruits echo orange

git commit --author="Greg <greg>" -am "split fruits from foods"

git checkout -
```

Repeat for the other files you want to split out. Let's say we also want to split out the veggies.

```
git checkout -b f2v

git mv foods veggies
git commit --author="Greg <greg>" -m "create veggies from foods"

>foods echo apple
>>foods echo cheese
>>foods echo eggs
>>foods echo grape
>>foods echo milk
>>foods echo orange
git add foods

>veggies echo celery
>>veggies echo lettuce
>>veggies echo peas

git commit --author="Greg <greg>" -am "split veggies from foods"

git checkout -
```

Then we octopus the branches together. However, the octopus will fail because the changes don't merge cleanly, so we'll have to do a manual octopus, like we did before.

```
>foods echo cheese
>>foods echo eggs
>>foods echo milk

>fruits echo apple
>>fruits echo grape
>>fruits echo orange

>veggies echo celery
>>veggies echo lettuce
>>veggies echo peas

git add foods fruits veggies
git write-tree
```

The `git write-tree` will emit a tree that represents the state of the index. We set up the index so that it contains the desired final state: The fruits have been put into `fruits`, the veggies into `veggies`, and the leftovers stay in `foods`.

Now to do the manual octopus merge.

```
git commit-tree ⟨tree-hash⟩ -p HEAD -p f2f -p f2v -m "split out fruits and veggies
from foods"
```

The `git commit-tree` will print a hash. This is the commit that is the result of the octopus merge. We can fast-forward to it.

```
git merge --ff-only ⟨commit-hash⟩
```

Okay, let's see what we ended up with.

```
git blame fruits

^e7a114d foods (Alice 2019-09-16 07:00:00 -0700 1) apple
86348be4 foods (Bob   2019-09-16 07:00:01 -0700 2) grape
34eb5bd1 foods (Carol 2019-09-16 07:00:02 -0700 3) orange

git blame veggies

^e7a114d foods (Alice 2019-09-16 07:00:00 -0700 1) celery
86348be4 foods (Bob   2019-09-16 07:00:01 -0700 2) lettuce
34eb5bd1 foods (Carol 2019-09-16 07:00:02 -0700 3) peas

git blame foods

^e7a114d (Alice 2019-09-16 07:00:00 -0700 1) cheese
86348be4 (Bob   2019-09-16 07:00:01 -0700 2) eggs
34eb5bd1 (Carol 2019-09-16 07:00:02 -0700 3) milk
```

Next time, we'll look at how to do this the easy way.

Raymond Chen

Follow