How to split out pieces of a file while preserving git line history: The easy way with misleading commits

devblogs.microsoft.com/oldnewthing/20190918-00

September 18, 2019

Raymond Chen

Last time, we split pieces of a file into separate files while preserving line history. We had to do some `git commit-tree` magic to get the results we wanted. But there's a way to do this with an octopus merge. You just have to make sure to keep the octopus happy.

Again, let's use the same scratch repo as we did for the last few days. 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.

To do things the easy way, we create a branch for each file we want to split out.

```
git checkout -b 2f

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

As before, 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 edit the `fruits` file so that it contains the lines we want to split out from `foods` (so far so good), but we also regenerate the `foods` file with only its *final contents*. We intend to delete the vegetables from the `foods` file, so we'll delete *both* the fruits *and* the vegetables. and the rest go back into the `foods` file.

```
>foods echo cheese
>>foods echo eggs
>>foods echo milk
git add foods

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

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

git checkout -
```

This is *completely misleading* and looks like we've lost our minds. We are ostensibly splitting the fruits out from the foods, but we also *threw away the veggies*. Somebody looking at this commit in isolation will say, "Hey, what happened to all the veggies?"

But let's keep going. Repeat the exercise by splitting out the veggies and throwing away the fruit.

```
git checkout -b 2v

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

git checkout 2f foods

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

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

git checkout -
```

To save ourselves some typing, we used a `git checkout 2f foods` to say "Just give me the copy of `foods` from the `2f` branch."

Finally, on the main branch, we also edit the `foods` file into its final form.

```
git checkout 2f foods
git commit --author="Greg <greg>" -am "split fruits and veggies from foods"
```

This commit is also absurdly misleading because most of the contents of the `foods` file simply vanished!

```
git merge 2f 2v

Trying simple merge with 2f
Trying simple merge with 2v
Merge made by the 'octopus' strategy.
 fruits  | 3 +++
 veggies | 3 +++
 2 files changed, 6 insertions(+)
 create mode 100644 fruits
 create mode 100644 veggies
```

The result is now that all three files are at their desired final forms, with the desired final line attributions.

```
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
```

However, the way we got there is very strange, and includes quite a few *extremely misleading* commits. I don't really recommend it. I recommend doing it the hard way with `git commit-tree`. Yes, it's harder, but it's also much less misleading to people who come to the repo later.

Bonus chatter: The misleading commit on the main branch is necessary because of another bug in octopus merges: It silently ignores the `--no-ff` flag. You can see this if you skip the extra commit on the main branch and try an octopus merge right away:

```
git merge --no-ff 2f 2v

Fast-forwarding to: 2f
Trying simple merge with 2v
Merge made by the 'octopus' strategy.
 foods   | 6 ------
 fruits  | 3 +++
 veggies | 3 +++
 3 files changed, 6 insertions(+), 6 deletions(-)
 create mode 100644 fruits
 create mode 100644 veggies
```

Even though we said `--no-ff`, the octopus merge fast-forwarded anyway. The result is that the `foods` file failed to preserve line history.

```
git blame foods

61bca29b (Greg 2019-09-18 07:00:00 -0700 1) cheese
61bca29b (Greg 2019-09-18 07:00:00 -0700 2) eggs
61bca29b (Greg 2019-09-18 07:00:00 -0700 3) milk
```

All the lines got blamed on Greg, when they really should be blamed on Alice, Bob, and Carol.

Raymond Chen

Follow