化解冲突：git merge 与 git rebase 中的 ours 和 theirs

2017-02-16 | tutorial

今天聊一些 git 的高级话题：化解冲突、 `ours` 和 `theirs` 的用法，以及这两个代词在 `git merge` 与 `git rebase` 中的不同。

git merge 与 git rebase 的区别

先复习一下 `git merge` 与 `git rebase` 的区别。假设当前项目有两个分支 `local` 和 `upstream`，且工作环境处于 `local` 分支，如下图所示。

```
1              HEAD
2               |
3   local        C
4               /
5   upstream   A -- B
```

执行 `git merge upstream` 的结果是这样的：

```
1                   HEAD
2                    |
3   local          C -- D
4               /     /
5   upstream   A -- B
```

而执行 `git rebase upstream` 的结果略有区别：

```
1                   HEAD
2                    |
3   local             C'
4                   /
5   upstream   A -- B
```

`git merge` 会分别提取 B 和 C 不同于 A 的部分，将他们合并到一起，因此生成的结果 D 有两个祖先。但是，`git rebase` 的做法是找到从 A 到 B 的修改，并将这些修改作用于 C 上，因此不会产生额外的 commit D。

在实际的项目里，由于多重祖先会产生一些难以解决的问题，我们倾向于使用 `git rebase` 而少用 `git merge`.

冲突与化解

在上面的项目中，如果 B 和 C 修改了 A 的同一行代码，就会引入一个冲突。冲突必须手工解决。假设项目中有文件 `word.txt`，在版本 A, B 和 C 中的内容分别是 "astronauts", "boycott", "chocolate". 执行 merge 之后 `word.txt` 会变成下面的样子。

```
<<<<<<< HEAD
chocolate
=======
boycott
>>>>>>> upstream
```

rebase 得到的结果类似但并不相同。

```
<<<<<<< HEAD
boycott
=======
chocolate
>>>>>>> C
```

可以看出，`<<<<<<<` 和 `>>>>>>>` 清晰地标示出了冲突的部分。你需要将它们改成期望的样子。如果你想保留两个版本中的一个而不引入新的修订，就可以使用两个预先定义的代词。 `<<<<<<<` 和 `=======` 之间的部分称为 `ours`，`=======` 和 `>>>>>>>` 之间则称为 `theirs`. 此时，你只需要键入合适的指令即可化解冲突，而不需要进入文本编辑器。在当前项目可以得到如下运行结果：

```
# merge
git checkout --ours word.txt    # => chocolate
git checkout --theirs word.txt  # => boycott

# rebase
git checkout --ours word.txt    # => boycott
git checkout --theirs word.txt  # => chocolate
```

发现了什么不对劲的地方了么？没错！这是迄今为止 git 让我最困惑的一点：merge 和 rebase 对于 ours 和 theirs 的定义是完全相反的。在 merge 时，ours 指代的是当前分支，theirs 代表需要被合并的分支。而在 rebase 过程中，ours 指向了修改参考分支，theirs 却是当前分支。如果你仔细探究其中的原因，会发现 rebase 隐含了一个 `git checkout upstream` 的过程，将 `HEAD` 从 `local` 分支变成了 `upstream` 分支。

```
local        C
             /
upstream   A -- B
                |
               HEAD (during rebase process)
```

git 会在 rebase 结束后撤销这个改变，但它已经不可避免地影响了冲突的状态，使 rebase 中 ours 和 theirs 的定义与 merge 截然相反。因此，在使用 ours 与 theirs 时请格外小心。

批量化解冲突

最近，我被安排去处理一个巨大的 rebase 冲突——受影响的文件数以百计。即便是对每个文件执行 `git checkout --ours` 和 `git checkout --theirs` 也是一件痛苦的工作。为了避免重复劳动，我写了 一个批量处理脚本（需要科学上网）。这个脚本根据设定的规则自动执行指令，让我在三分钟之内完成了一个14步（以 `git rebase --continue` 计）的冲突化解。如果你有需要，欢迎 star 和 fork.