**遇到冲突时的分支合并**

有时候合并操作不会如此顺利。 如果你在两个不同的分支中，对同一个文件的同一个部分进行了不同的修改，Git 就没法干净的合并它们。 如果你对 #53 问题的修改(也就是在iss53分支上有修改)和有关 hotfix 的修改(也就是在hotfix分支上的修改)都涉及到同一个文件的同一处，在合并它们的时候就会产生合并冲突：

$ git merge iss53

Auto-merging index.html

CONFLICT (content): Merge conflict in index.html

Automatic merge failed; fix conflicts and then commit the result.

此时 Git 做了合并，但是没有自动地创建一个新的合并提交。 Git 会暂停下来，等待你去解决合并产生的冲突。 你可以在合并冲突后的任意时刻使用 git status 命令来查看那些因包含合并冲突而处于未合并（unmerged）状态的文件：

$ git status

On branch master

You have unmerged paths.

(fix conflicts and run "git commit")

Unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: index.html

no changes added to commit (use "git add" and/or "git commit -a")

任何因包含合并冲突而有待解决的文件，都会以未合并状态标识出来。 Git 会在有冲突的文件中加入标准的冲突解决标记，这样你可以打开这些包含冲突的文件然后手动解决冲突。 出现冲突的文件会包含一些特殊区段，看起来像下面这个样子：

<<<<<<< HEAD:index.html

<div id="footer">contact : email.support@github.com</div>

=======

<div id="footer">

please contact us at support@github.com

</div>

>>>>>>> iss53:index.html

这表示 HEAD 所指示的版本（也就是你的 master 分支所在的位置，因为你在运行 merge 命令的时候已经检出到了这个分支）在这个区段的上半部分（======= 的上半部分），而 iss53 分支所指示的版本在 ======= 的下半部分。 为了解决冲突，你必须选择使用由 ======= 分割的两部分中的一个，或者你也可以自行合并这些内容。 例如，你可以通过把这段内容换成下面的样子来解决冲突：

<div id="footer">

please contact us at email.support@github.com

</div>

上述的冲突解决方案仅保留了其中一个分支的修改，并且 <<<<<<< , ======= , 和 >>>>>>> 这些行被完全删除了。 ***在你解决了所有文件里的冲突之后，对每个文件使用 git add 命令来将其标记为冲突已解决***。***Git add以后会将冲突解决的文件放入index, 然后再等待commit. 一旦暂存这些原本有冲突的文件，Git 就会将它们标记为冲突已解决。***

如果你想使用图形化工具来解决冲突，你可以运行 git mergetool，该命令会为你启动一个合适的可视化合并工具，并带领你一步一步解决这些冲突：

$ git mergetool

This message is displayed because 'merge.tool' is not configured.

See 'git mergetool --tool-help' or 'git help config' for more details.

'git mergetool' will now attempt to use one of the following tools:

opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmerge p4merge araxis bc3 codecompare vimdiff emerge

Merging:

index.html

Normal merge conflict for 'index.html':

{local}: modified file

{remote}: modified file

Hit return to start merge resolution tool (opendiff):

如果你想使用除默认工具（在这里 Git 使用 opendiff 做为默认的合并工具，因为作者在 Mac 上运行该程序）外的其他合并工具，你可以在 “下列工具中（one of the following tools）” 这句后面看到所有支持的合并工具。 然后输入你喜欢的工具名字就可以了。

|  |  |
| --- | --- |
| Note | 如果你需要更加高级的工具来解决复杂的合并冲突，我们会在 [高级合并](https://git-scm.com/book/zh/v2/ch00/r_advanced_merging) 介绍更多关于分支合并的内容。 |

等你退出合并工具之后，Git 会询问刚才的合并是否成功。 如果你回答是，Git 会暂存那些文件以表明冲突已解决： 你可以再次运行 git status 来确认所有的合并冲突都已被解决：

情景1:

如果本地分支上有A,B,C三个文件。 其中B,C文件在本地修改但没有提交，而A文件没有修改。 然后我们在另一个branch上修改了A文件，然后merge到本分支的时候，那么不会有冲突，A文件会有merge的改变，而B, C文件将没有因为merge而产生改变。Git status显示状态是B, C文件modified but not added to commit.

情景2:

如果本地分支上有A,B,C三个文件。 其中A文件在本地修改但没有提交。 然后我们在另一个branch上修改了A文件，然后merge到本分支的时候，那么git将提示merge的分支将会取消我们所在分支对A文件的修改，需要stash以后才可以merge.

情景3

如果本地分支上有A,B,C三个文件。 其中A文件在本地修改并且提交。 然后我们在另一个branch上修改了A文件，然后merge到本分支的时候，那么如果另一个分支的A文件和我们本地提交A文件的修改没有冲突，那么我们将合并在一起。如果有冲突，那么A文件的内容将会有冲突标记（这也是为什么git fetch + merge比git pull安全的原因。Git fetch以后可以先通过git diff来查看以后merge以后是否有冲突，而git pull直接就合并，直接会在文件中产生冲突标记），需要我们手动解决冲突以后再本地提交，再push一次,然后提交记录中将会存在merge进来的额外的提交记录。