分支与合并基础

一个Git仓库可以维护很多开发分支。现在我们来创建一个新的叫”experimental”的分支：

$ git branch experimental

如果你运行下面这条命令：

$ git branch

你会得到当前仓库中存在的所有分支列表：

experimental

\* master

“experimental” 分支是你刚才创建的，“master”分支是Git系统默认创建的主分支。星号(“\*”)标识了你当工作在哪个分支下，输入：

$ git checkout experimental

切换到”experimental”分支，先编辑里面的一个文件，再提交(commit)改动，最后切换回 “master”分支。

(edit file)

$ git commit -a

$ git checkout master

你现在可以看一下你原来在“experimental”分支下所作的修改还在不在；因为你现在切换回了“master”分支，所以原来那些修改就不存在了。

你现在可以在“master”分支下再作一些不同的修改:

(edit file)

$ git commit -a

这时，两个分支就有了各自不同的修改(diverged)；我们可以通过下面的命令来合并“experimental”和“master”两个分支:

$ git merge experimental

如果这个两个分支间的修改没有冲突(conflict), 那么合并就完成了。如有有冲突，输入下面的命令就可以查看当前有哪些文件产生了冲突:

$ git diff

当你编辑了有冲突的文件，解决了冲突后就可以提交了：

$ git commit -a

提交(commit)了合并的内容后就可查看一下:

$ gitk

执行了gitk后会有一个很漂亮的图形的显示项目的历史。

这时你就可以删除掉你的 “experimental” 分支了(如果你愿意)：

$ git branch -d experimental

git branch -d只能删除那些已经被当前分支的合并的分支. 如果你要强制删除某个分支的话就用git branch –D；下面假设你要强制删除一个叫”crazy-idea”的分支：

$ git branch -D crazy-idea

分支是很轻量级且容易的，这样就很容易来尝试它。

如何合并

你可以用下面的命令来合并两个分离的分支：[git merge](http://www.kernel.org/pub/software/scm/git/docs/git-merge.html):

$ git merge branchname

这个命令把分支"branchname"合并到了当前分支里面。如有冲突(冲突--同一个文件在远程分支和本地分支里按不同的方式被修改了）；那么命令的执行输出就像下面一样

$ git merge next

100% (4/4) done

Auto-merged file.txt

CONFLICT (content): Merge conflict in file.txt

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

在有问题的文件上会有冲突标记，在你手动解决完冲突后就可以把此文件添加到索引(index)中去，用git commit命令来提交，就像平时修改了一个文件一样。

如果你用gitk来查看commit的结果，你会看到它有两个父分支：一个指向当前的分支，另外一个指向刚才合并进来的分支。

解决合并中的冲突

如果执行自动合并没有成功的话，git会在索引和工作树里设置一个特殊的状态， 提示你如何解决合并中出现的冲突。

有冲突(conflicts)的文件会保存在索引中，除非你解决了问题了并且更新了索引，否则执行 [git commit](http://www.kernel.org/pub/software/scm/git/docs/git-commit.html)都会失败:

$ git commit

file.txt: needs merge

如果执行 [git status](http://www.kernel.org/pub/software/scm/git/docs/git-status.html) 会显示这些文件没有合并(unmerged),这些有冲突的文件里面会添加像下面的冲突标识符:

<<<<<<< HEAD:file.txt

Hello world

=======

Goodbye

>>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt

可以看到 ======= 隔开的上半部分，是 HEAD（即 master 分支，在运行 merge 命令时检出的分支）中的内容，下半部分是在合并的分支中的内容。解决冲突的办法无非是从二者选其一或者由你亲自整合到一起。

你所需要的做是就是编辑解决冲突，（接着把冲突标识符 <<<<<<<，=======，和>>>>>>> 这些行删掉），再执行下面的命令:

$ git add file.txt

$ git commit

注意：提交注释里已经有一些关于合并的信息了，通常是用这些默认信息，但是你可以添加一些你想要的注释。

上面这些就是你要做一个简单合并所要知道的，但是git提供更多的一些信息来 帮助解决冲突。

撒销一个合并

如果你觉得你合并后的状态是一团乱麻，想把当前的修改都放弃，你可以用下面的命令回到合并之前的状态：

$ git reset --hard HEAD

或者你已经把合并后的代码提交，但还是想把它们撒销：

$ git reset --hard ORIG\_HEAD

但是刚才这条命令在某些情况会很危险，如果你把一个已经被另一个分支合并的分支给删了，那么 以后在合并相关的分支时会出错。

快速向前合并

还有一种需要特殊对待的情况，在前面没有提到。通常，一个合并会产生一个合并提交(commit), 把两个父分支里的每一行内容都合并进来。

但是，如果当前的分支和另一个分支没有内容上的差异，就是说当前分支的每一个提交(commit)都已经存在另一个分支里了，git 就会执行一个“快速向前"(fast forward)操作；git 不创建任何新的提交(commit),只是将当前分支指向合并进来的分支。

### Git Community Book 中文版

<http://gitbook.liuhui998.com/>