

git merge 合并分支

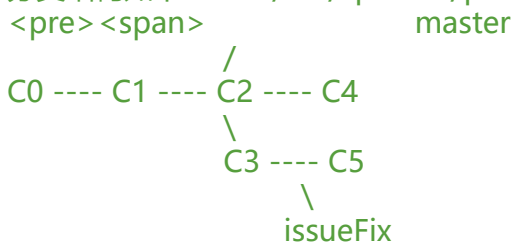
作者: [xiaoD](#)

原文链接: <https://ld246.com/article/1467602044460>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

[Git](http://lib.csdn.net/base/28 "Git知识库") merge 用来做分支合并，将其他分支中的内容合并到当前分支中。比分支结构如下：



当前分支是master
\$ git checkout master

把issueFix中的内容Merge进来：
\$ git merge issueFix

如果没有冲突的话，merge完成。有冲突的话，git会提示那个文件中有冲突，比如有如冲突：

<<<<<<<< HEAD:test.c

printf (“test1″);

====

printf (“test2″);

>>>>>>> issueFix:test.c

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 issueFix 分支中的内容。解决冲突的办法无非是二者选其一者由你亲自整合到一起。比如你可以通过把这段内容替换为下面这样来解决：

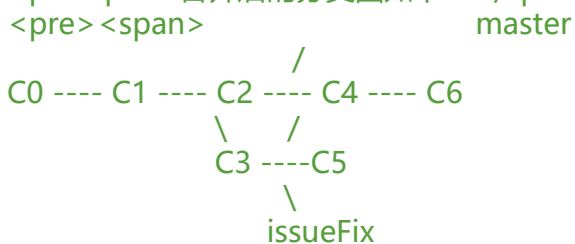
printf (“test2″);

这个解决方案各采纳了两个分支中的一部分内容，而且删除了 <<<<<<<< ;, =====, 和>>>>>>> 这些行。在解决了所有文件里的所有冲突后，行 git add 将它们标记为已解决（resolved）。因为一旦暂存，就表示冲突已经解决。如果你想用一个有图形界面的工具来解决这些问题，不妨运行 git mergetool，它会调用一个可视化的合并工具并引导你解决所有冲突：

\$ git mergetool
merge tool candidates: kdiff3 tkdiff xxdiff meld gvimdiff oendiff emerge vimdiff
Merging the files: index.html

Normal merge conflict for ‘test.c’;
{local}: modified
{remote}: modified
Hit return to start merge resolution tool (kdiff3):

合并后的分支图如下：



注意，这次合并的实现，由于当前 master 分支所指向的 commit (C4)并非想要并入分支 (issueFix) 的直接祖先，Git 不得不进行一些处理。就此例而言，Git 会用两个分支的末端 (C4 和 C) 和它们的共同祖先 (C2) 进行一次简单的三方合并。对三方合并的结果作一新的快照，并自动创建一个指向它的 commit (C6)

退出合并工具以后，Git 会询问你合并是否成功。如果回答是，它会为你把相关文件暂存起来，以表明状态为已解决。然后可以用 git commit 来完成这次合并提交。