快乐&&平凡

本博客所记录的文章，主要是从网络收集的，有一些因为经过多次转载，所以出处已经不知，若是侵权，请通知我，我及时修改。本博客主要是用来记录我对所写文章的理解，若有错误，请大家指点，相互学习！

个人资料

wh_19910525

访问： 1254793次
积分： 15167
等级： BLOG 7
排名： 第320名

原创： 363篇 转载： 94篇
译文： 2篇 评论： 215条

git merge 和 git 吉

分类： git 详解 常用的 git 命 2012-05-10 16:31 17421人阅读 评论(3) 收藏 举报

merge git

git merge是用来合并两个分支的。

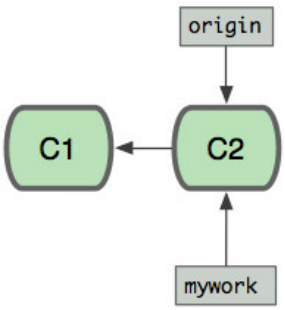
git merge b

将b分支合并到当前分支

同样 git rebase b，也是把 b分支合并到当前分支

他们的 原理 如下：

假设你现在基于远程分支"origin"，创建一个叫"mywork"的分支。
$ git checkout -b mywork origin
假设远程分支"origin"已经有了2个提交，如图



现在我们在这个分支做一些修改，然后生成两个提交(commit).
$ vi file.txt
$ git commit
$ vi otherfile.txt
$ git commit
...
但是与此同时，有些人也在"origin"分支上做了一些修改并且做了提交了. 这就意味着"origin"和"mywork"这两个分支各自"前进"了，它们之间"分叉"了。

eclipse (16)
ubuntu (21)
next-day (3)
shell (18)
windows相关 (22)
java相关 (13)
android-app (32)
often_use (15)
pmu (1)
pychon 学习 (2)
Makefie (10)
pythen学习 (2)
QT (9)
android-jni (3)
anroid build system (15)

文章存档

2015年09月 (3)
2015年08月 (5)
2015年07月 (4)
2015年04月 (4)
2015年03月 (5)
展开

阅读排行

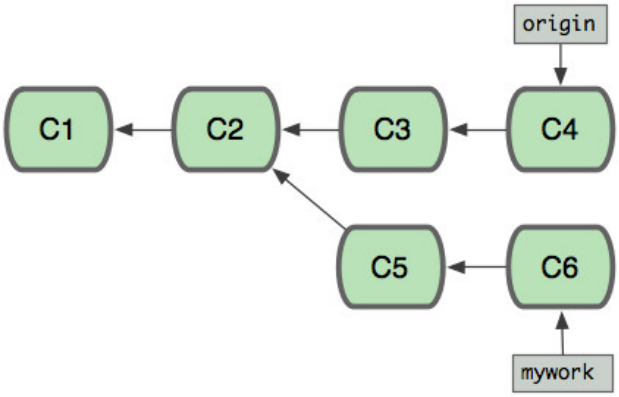
This Android SDK requir (50103)
git push 小结 (38531)
repo的小结 (37530)
git 使用详解（5）-- get l (30092)
boot.img的解包与打包 (27097)
ssh-keygen 的 详解 (26522)
apk反编译获取完整源码 (26465)
android:versionCode和a (22138)
git cherry-pick 小结 (20692)
bat批处理的注释语句 (19745)

评论排行

This Android SDK requir (61)
从 08年开始写博客，之前 (8)
AndroidManifest中origina (5)
Android 在 SElinux下 如何 (5)
apk反编译获取完整源码 (5)
如何在windows下使用git (4)
8 Android平台开发-WIFI (4)
#!/usr/bin/env python与# (4)
Android系统Recovery工作 (4)
git stash和git stash pop (4)

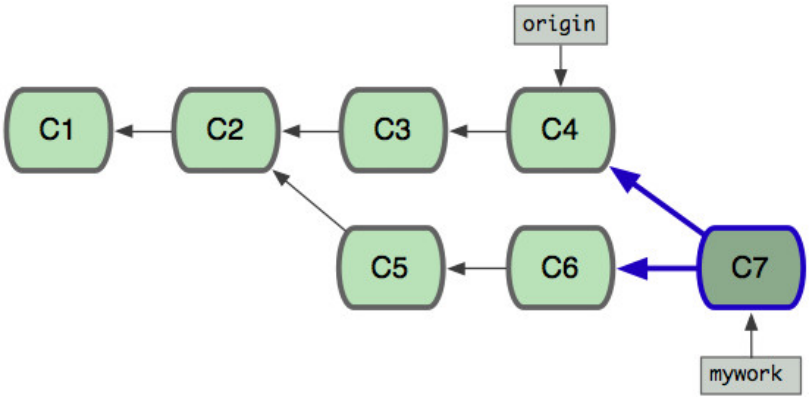
推荐文章

* 最老程序员创业开发实训4--IOS平台下MVC架构
* Android基础入门教程--6.1 数据存储与访问之--文件存储读写
* 聊天界面的制作（三）--表情列表发送功能



在这里，你可以用"pull"命令把"origin"分支上的修改拉下来并且和你的修改合并； 结果看起来就像一个新的"合并的提交"(merge commit):

git merge



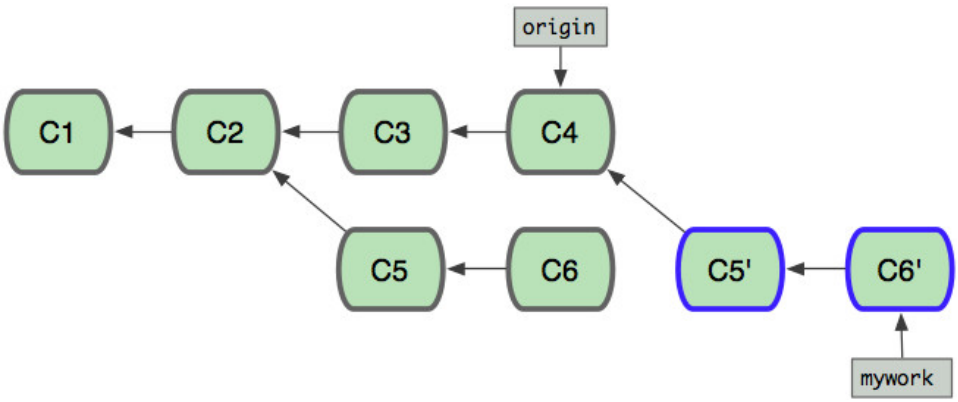
但是，如果你想让"mywork"分支历史看起来像没有经过任何合并一样，你也许可以用 git rebase:

$ git checkout mywork

$ git rebase origin

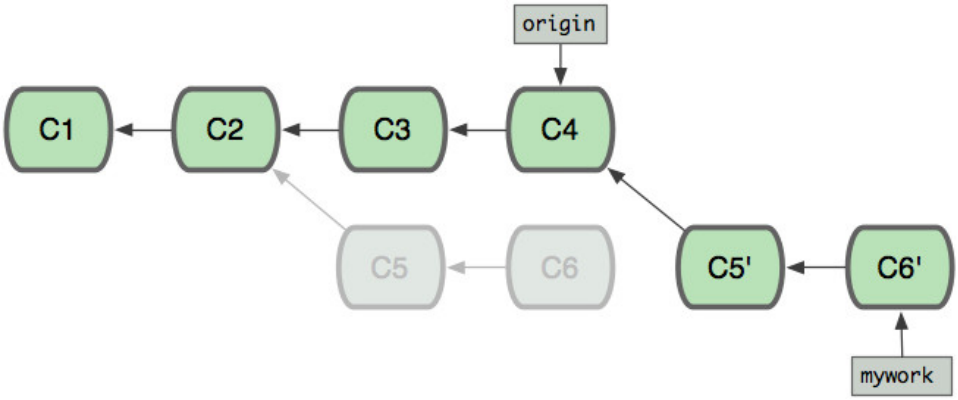
这些命令会把你的"mywork"分支里的每个提交(commit)取消掉，并且把它们临时 保存为补丁(patch)(这些补丁放到".git/rebase"目录中),然后把"mywork"分支更新 为最新的"origin"分支，最后把保存的这些补丁应用到"mywork"分支上。

git rebase



当'mywork'分支更新之后，它会指向这些新创建的提交(commit),而那些老的提交会被丢弃。 如果运行垃圾收集命令(pruning garbage collection), 这些被丢弃的提交就会删除. （请查看 git gc)

* Linux下编程--文件与IO（三）文件共享和fcntl函数
* Windows 多进程通信API总结
* libuv里的几个缺陷



二、解决冲突

在rebase的过程中，也许会出现冲突(conflict). 在这种情况，Git会停止rebase并会让你去解决 冲突；在解决完冲突后，用"git-add"命令去更新这些内容的索引(index), 然后，你无需执行 git-commit,只要执行:

```
$ git rebase --continue
```

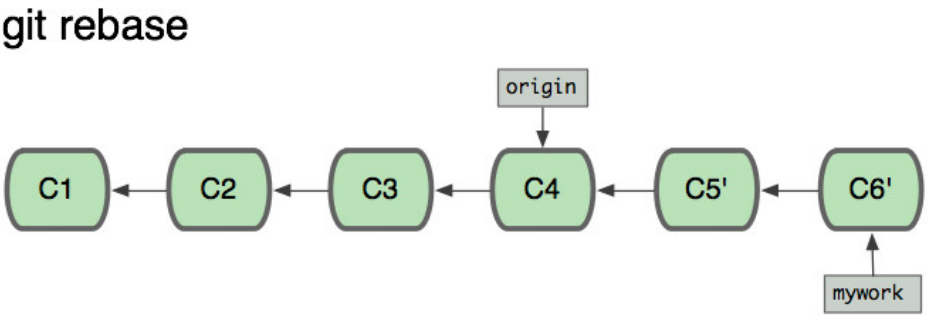
这样git会继续应用(apply)余下的补丁。

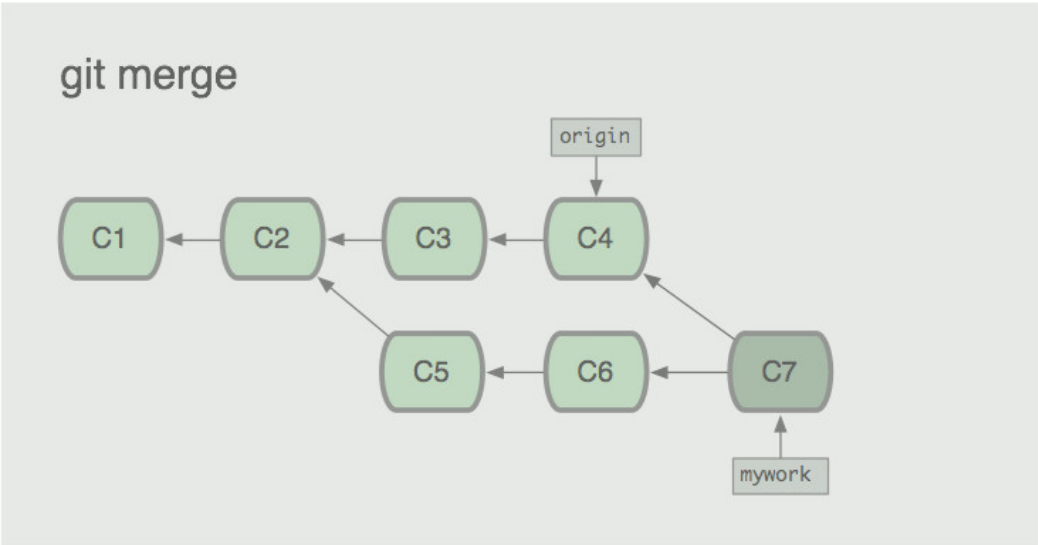
在任何时候，你可以用--abort参数来终止rebase的行动，并且"mywork" 分支会回到rebase开始前的状态。

```
$ git rebase --abort
```

三、git rebase和git merge的区别

现在我们可以看一下用合并(merge)和用rebase所产生的历史的区别:





当我们使用Git log来参看commit时，其commit的顺序也有所不同。

假设C3提交于9:00AM,C5提交于10:00AM,C4提交于11:00AM，C6提交于12:00AM,

对于使用git merge来合并所看到的commit的顺序（从新到旧）是：C7 ,C6,C4,C5,C3,C2,C1

对于使用git rebase来合并所看到的commit的顺序（从新到旧）是：C7 ,C6',C5',C4,C3,C2,C1

因为C6'提交只是C6提交的克隆，C5'提交只是C5提交的克隆，

从用户的角度看使用git rebase来合并后所看到的commit的顺序（从新到旧）是：C7 ,C6,C5,C4,C3,C2,C1