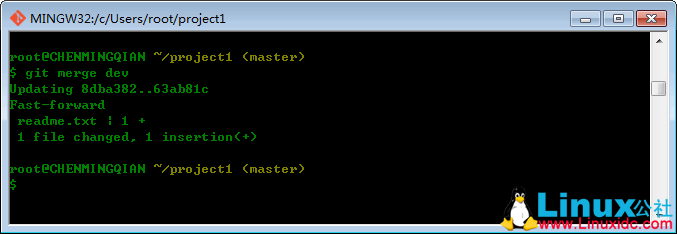
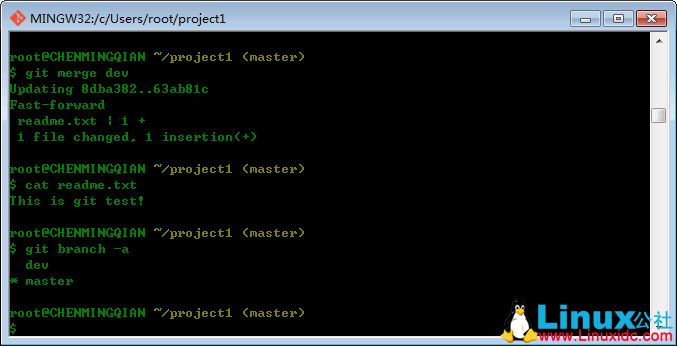
4.合并分支（快速合并）

现在，我们把dev分支的工作成果合并到master分支上，如下图：



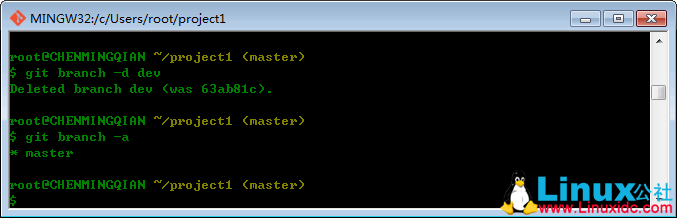
git merge 命令用于合并指定分支到当前分支。合并后，再查看readme.txt的内容，就可以看到，和dev分支的最新提交是完全一样的。如下图：



大家注意到上面的Fast-forward信息，Git告诉我们，这次合并是“快进模式”，也就是直接把master指向dev的当前提交，所以合 并速度非常快。当然，也不是每次合并都能Fast-forward，我们后面会将其他方式的合并。合并完成后，就可以放心地删除dev分支了。

5.删除分支

下面我们来演示一下删除分支，如下图：

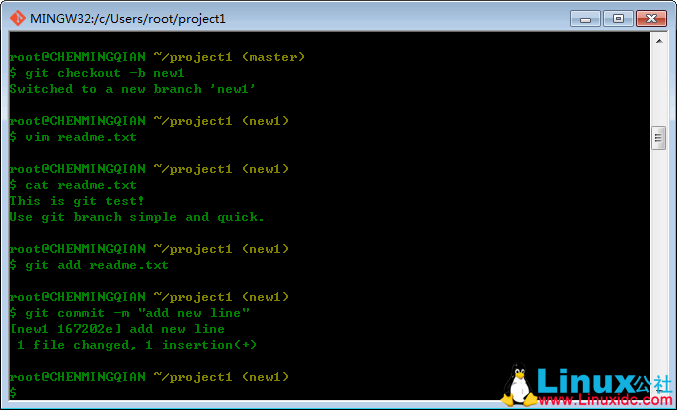


大家在实际操作中可以发现在我们创建、合并、删除分支的速度非常快吧，这和直接在master分支上工作效果是一样的，但过程更安全，更可靠。下面我们来简单的总结一下：

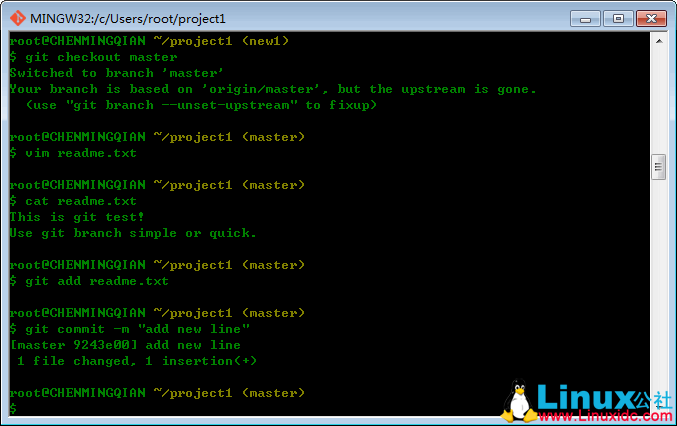
* 查看分支 git branch -a
* 创建分支 git branch name
* 切换分支 git checkout name
* 创建并切换 git checkout -b name
* 合并某分支到当前分支 git merge name
* 删除分支 git branch -d name

6.分支合并冲突

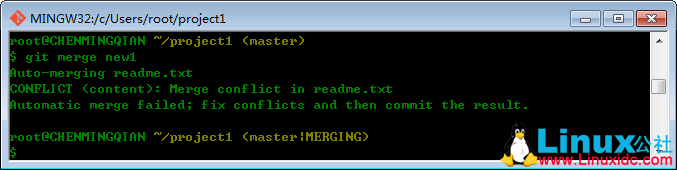
下面我们来演示一下不同分支修改同一个文件产生冲突问题，下面我们来创建一个新的分支，继续开发新功能：



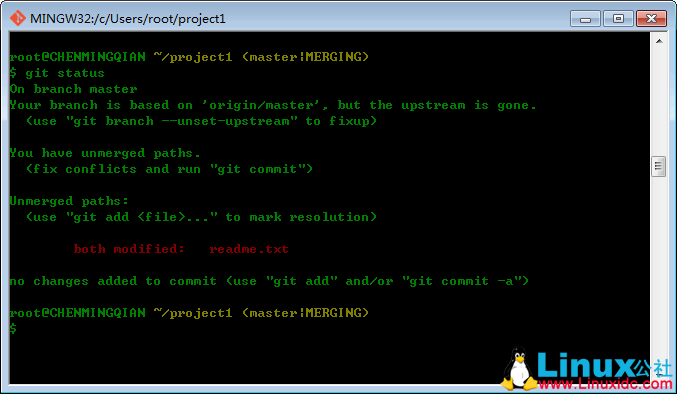
现在我们将分支切换到master分支上修改readme.txt内容并提交，如下图：



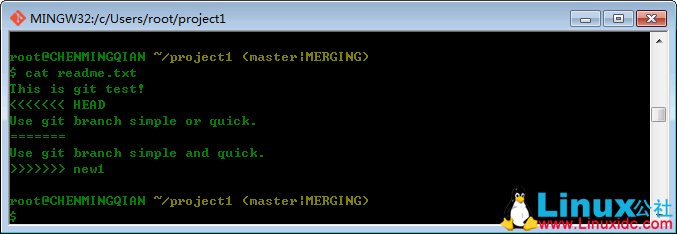
这种情况下，Git无法执行“快速合并”，只能试图把各自的修改合并起来，但这种合并就可能会有冲突，如下图：



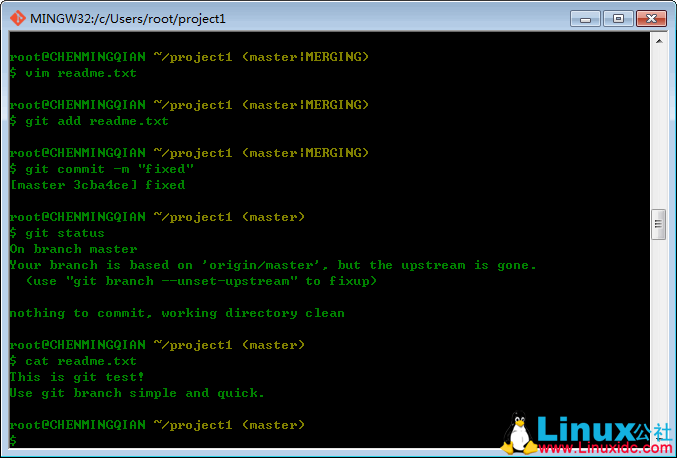
果然冲突了，Git告诉我们readme.txt文件存在冲突，必须手动解决冲突后再提交。下面我们用git status查看一下状态：



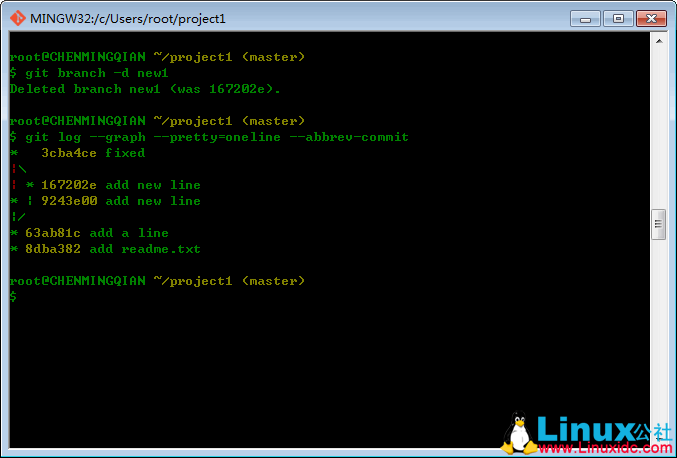
下面我们来查看一下readme.txt中的内容，如下图：



Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容，让我们选择要保留的内容，下面我们修改一下readme.txt，再次提交。如下图：



好了，到这我们的分支合并冲突就讲解完成了，下面我们来删除分支。如下图：



最后，我们可以用 git log --graph --pretty=oneline --abbrev-commit 命令，查看一下分支合并。