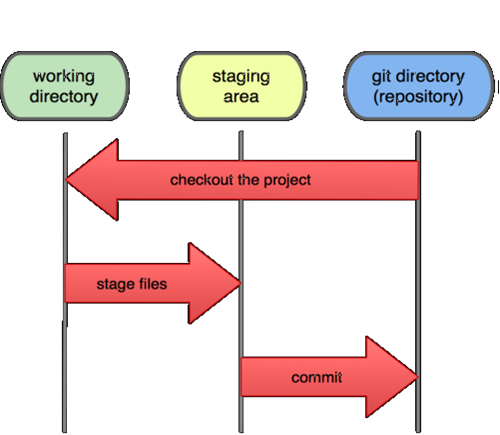
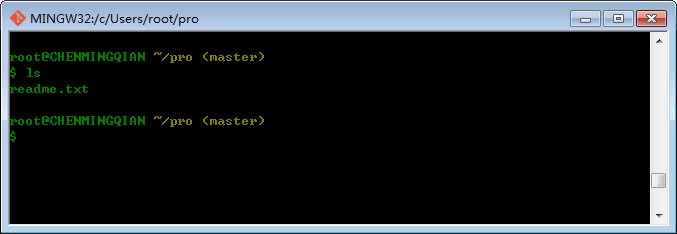
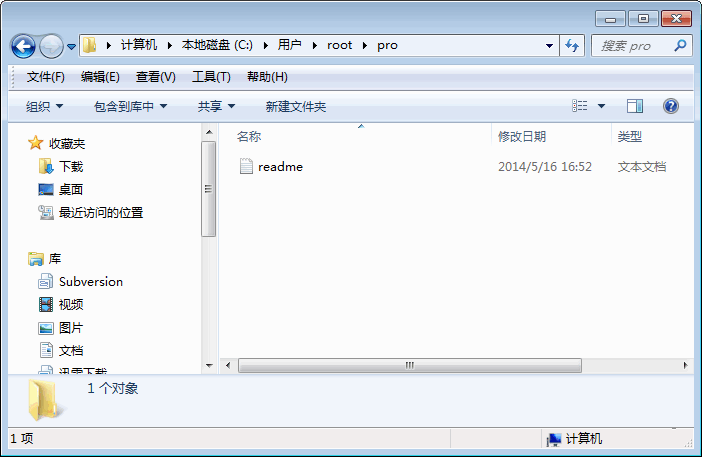
3.工作区、提交区/暂存区（stage/index）、版本库

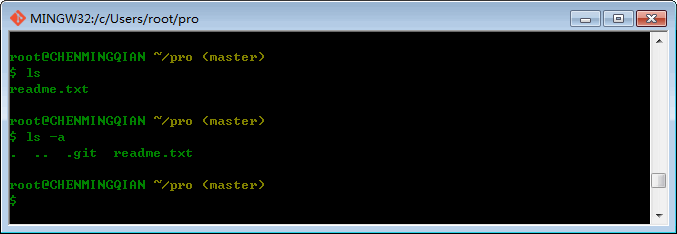


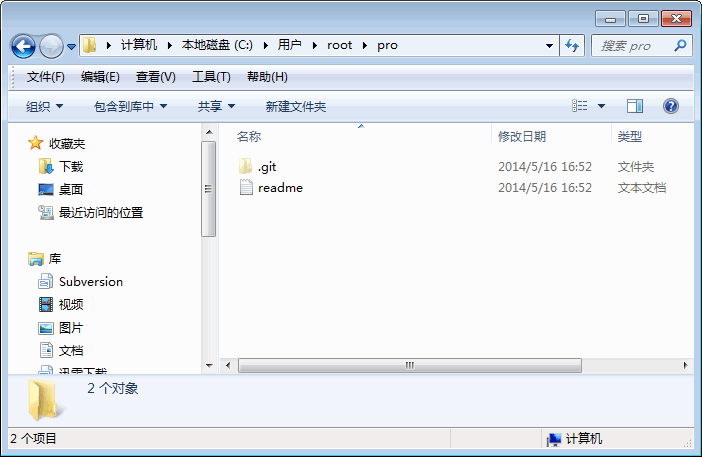
其实呢，工作区、提交区/暂存区（stage/index）、版本库的概念问题，从上图中就能看的很清楚，本来不想细讲的，但想想还是说一下。 Git与其他版本版本控制器其中之一的不同之处就在于有提交区/暂存区（stage/index）的概念。下面我们先来看一下工作区：



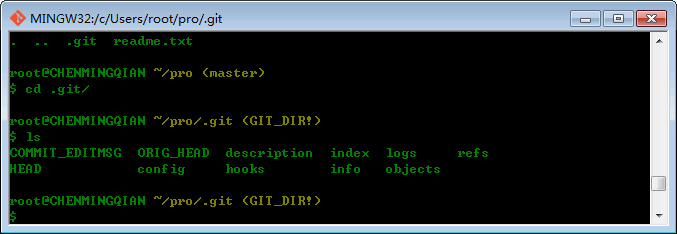


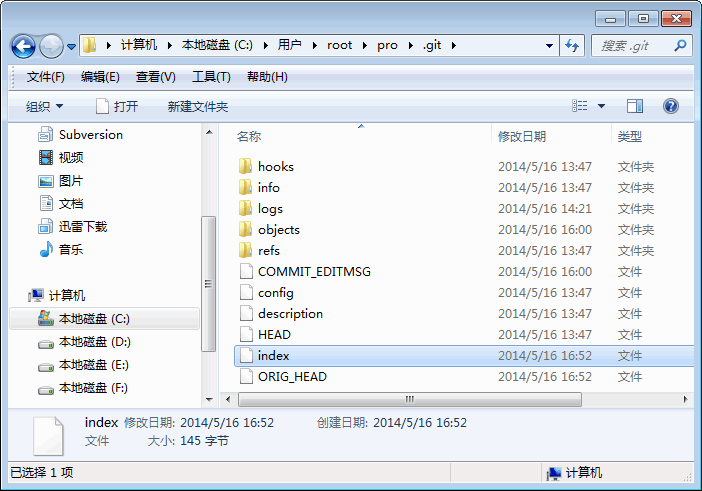
其实呢，工作区就是我们开发目录了，在电脑中是可能看到的，比如我们这里的pro目录，就是一个工作区。大家再来看一下，下面的两张图：





大家可以看到，工作区中有个隐藏的目录“.git”，这个不是工作区哦，这个就是Git的版本库。大家再看下面两张图：

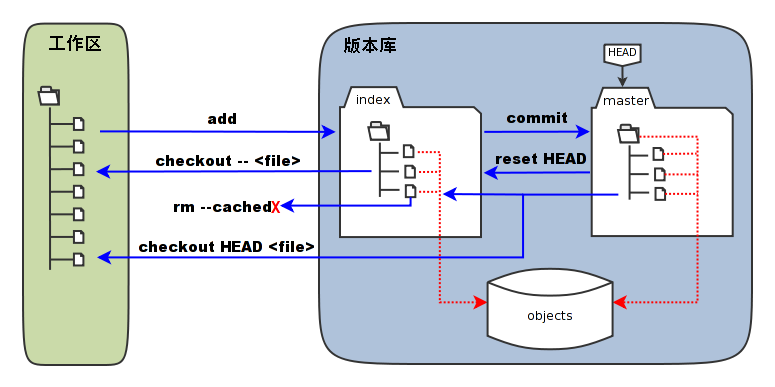




大家可以看到，在“.git”目录中有很多文件，其中一个重要的文件index，就是我们说的提交区/暂存区（stage/index）。暂存区 （stage, index）是 Git 最重要的概念之一，理解了这个概念很多 Git 命令就不再那么神秘了。对于 Git 暂存区（stage） ，不知道您的感想如何？

* “被眼花缭乱的 Git 魔法彻底搞糊涂了？”
* “Git 为什么这么折磨人，修改的文件直接提交不就完了么？”
* “看不出 Git 这么做有什么好处？”

我认为 Git 暂存区（stage或称为 index）的设计是 Git 最成功的设计之一，也是最难理解的一个设计。 在版本库（.git）目录下，有一个 index 文件，相信大家在上图中已经看到了。下面我们好好说一说他们之间关系，同样的我们先看一张图：



在上图中，我们可以看到部分 Git 命令是如何影响工作区和暂存区（stage/index）的。

* 图中左侧为工作区，右侧为版本库。在版本库中标记为 "index" 的区域是暂存区（stage/index），标记为 "master" 的是 master 分支所代表的目录树（关于分支问题在下面的文章中会详解）。
* 图中我们可以看出此时 "HEAD" 实际是指向 master 分支的一个“指针”。所以，图示的命令中出现 HEAD 的地方可以用 master 来替换（HEAD的概念我们在后面的文章中也会详解）。
* 图中的 objects 标识的区域为 Git 的对象库，实际位于 ".git/objects" 目录下，我们会在后面的文章中将重点介绍，嘿嘿！。
* 当对工作区新增或修改的文件执行 "git add" 命令时，暂存区的目录树被更新，同时工作区新增或修改的文件内容被写入到对象库中的一个新的对象中，而该对象的ID被记录在暂存区的文件索引中。（如上图）
* 当执行提交操作 "git commit" 时，暂存区的目录树写到版本库的对象库（objects）中，master 分支会做相应的更新。即 master 指向的目录树就是提交时暂存区的目录树。（如上图）
* 当执行 "git reset HEAD" 命令时，暂存区的目录树会被重写，被 master 分支指向的目录树所替换，但是工作区不受影响。 当执行 "git rm --cached <file>" 命令时，会直接从暂存区删除文件，工作区则不做出改变。
* 当执行 "git checkout ." 或者 "git checkout -- <file>" 命令时，会用暂存区全部或指定的文件替换工作区的文件。这个操作很危险，会清除工作区中未添加到暂存区的改动。
* 当执行 "git checkout HEAD ." 或者 "git checkout HEAD <file>" 命令时，会用 HEAD 指向的 master 分支中的全部或者部分文件替换暂存区和以及工作区中的文件。这个命令也是极具危险性的，因为不但会清除工作区中未提交的改动，也会清除暂存区中未提交的改 动。

好了，到这里我们的工作区、暂存区、版本库就讲解到这里了，由于本人能力有限有什么不正确的地方欢迎大家指出。好了，下面我们继续讲解……

[linux](http://www.linuxidc.com/)