一，本地库操作

1，本地库初始化

q：退出当前进入的目录；

ll：列出当前目录下的文件的详细情况列出来；

ls -lA：查看当前目录下的所有文件的详细情况，包括隐藏文件；

ls -l|less：管道操作，分屏管理；

mkdir 文件名：表示创建一个文件；

cd 文件夹：表示进入一个文件夹；

pwd：查看当前的路径；

（在Linux中，以.开头的文件目录都是影藏的资源，需要使用ls -lA才能看见）

ll .git/：表示查看.git下所有的文件信息；

（1）命令：`git init`

（2）效果：

```
Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/wechat (master)
$ ll .git/
total 7
-rw-r--r-- 1 Anly 197121 130 5月  24 21:00 config
-rw-r--r-- 1 Anly 197121  73 5月  24 21:00 description
-rw-r--r-- 1 Anly 197121  23 5月  24 21:00 HEAD
drwxr-xr-x 1 Anly 197121   0 5月  24 21:00 hooks/
drwxr-xr-x 1 Anly 197121   0 5月  24 21:00 info/
drwxr-xr-x 1 Anly 197121   0 5月  24 21:00 objects/
drwxr-xr-x 1 Anly 197121   0 5月  24 21:00 refs/
```

（3）注意事项：.git目录中存放的是本地库相关的子目录和文件，不要删除，也不要胡乱修改。

设置签名

（1）形式：

用户名：anqili

email地址：（email可以不是真实的邮箱地址）anqili@163.com

（2）作用

区分不同开发人员的身份。

（3）辨析：这里设置的签名和登录（代码托管中心）的账号和密码没有任何关系。

（4）设置签名的命令：

①项目级别（仓库级别）：仅在当前本地库范围内有效。项目级别的命令是：git config user.name 用户名；git config user.email 邮箱地址。

②系统用户级别：登录当前操作系统的用户范围，比项目级别的范围要大，默认使用系统用户级别，但是如果设置了项目级别的，则使用项目级别的。即就近原则：项目级别优先于系统用户级别，二者都有时，使用项目级别的签名。如果只有系统级别的签名，就以系统用户级别的签名为准。二者都没有是不允许的。 系统用户的命令是：git config --global user.name 用户名；git config --global user.email 邮箱地址。

a，项目级别的签名信息保存在.git目录下的config文件中。使用cat .git/config查看config中的详细信息。

```
Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/wechat (master)
$ git  config user.name anqili_pro

Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/wechat (master)
$ git config user.email anqili_pro@163.com

Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/wechat (master)
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = false
        bare = false
        logallrefupdates = true
        symlinks = false
        ignorecase = true
[user]
        name = anqili_pro
        email = anqili_pro@163.com
```

b，系统用户级别的签名信息保存在~/.gitconfig文件中：

```
Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/wechat (master)
$ git config --global user.username anqili1_global

Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/wechat (master)
$ git config --global user.email anqili2_global@163.com
```

用户家目录“~”

```
Anly@DESKTOP-2API2N9 MINGW64 ~
$ ls -less
ls: unknown option -- e
Try 'ls --help' for more information.

Anly@DESKTOP-2API2N9 MINGW64 ~
$ ls -la|less

Anly@DESKTOP-2API2N9 MINGW64 ~
$ cat .gitconfig
[user]
        username = anqili1_global
        email = anqili2_global@163.com
```

查看命令如下：

```
Anly@DESKTOP-2API2N9 MINGW64 ~
$ cat ~/.gitconfig
[user]
        username = anqili1_global
        email = anqili2_global@163.com
```

实际开发中采用系统用户级别的签名。

2，基本操作（git的专属命令都是以git开头的，即git是主命令，git后面的命令是子命令）

（1）git status：查看工作区或者缓存区或者本地库的一个状态。no commit yet表示在这个本地库中没有任何提交过的东西。可提交的东西放在缓存区。

```
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

track表示追踪，就是使用git来管理创建或者复制过来的文件。

对于需要创建的文件，可以事先在初始化的git本地库文件夹下将文件创建好，而不是在Git base下通过命令的方式创建，但是一定要注意，一定要将文件放在git本地仓库的文件夹下。

通过“git add 文件名”将一个工作区的文件加载到缓存区中去。工作区中的文件是红色的。缓存区中的文件是绿色的。

```
Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/WeChat (master)
$ git add test.txt

Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/WeChat (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.txt
```

通过“git rm --cached 文件名”表示将一个放在暂存区的文件撤回到工作区状态，并不会动工作区中的内容，只是从暂存区中将文件移除而

已。采用git status查看又变成红色的。

采用“git commit 文件名”将一个暂存区的文件提交到本地库中。（注意文件名需要带后缀名）

进入vim编辑状态的时候，按i键即可进行编辑，编辑结束之后，按ESC后再在命令行最后输入：wq即可退出vim编辑状态。

使用”cat 文件名+文件后缀“查看文件中的内容。

使用“git checkout --文件名+文件后缀”撤销对文件的修改。

使用“git commit -m ″本次需要提交的解释信息“ 文件名”表示提交文件到本地库中，并且不需要进入vim编辑器。

命令总结：

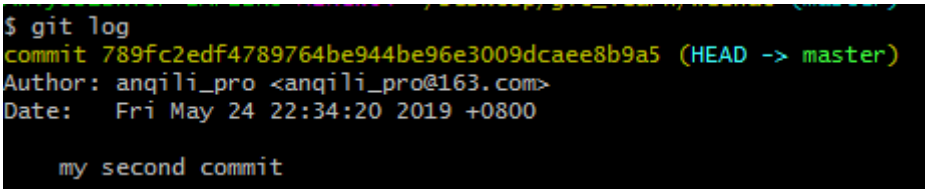
（1）状态查看操作：git status(查看工作区、暂存区、以及本地库的状态)

（2）添加操作：git add filename(将工作区的”新建/修改的文件添加到暂存区“)

（3）提交操作：git commit -m ″commit message″ filename(将暂存区的内容提交到本地库)

（4）查看历史记录的操作：（git log --oneline只能显示现在和以前的版本信息；但是git reflog能显示所有的版本信息）

①git log：以最完整的形式显示日志信息；（多屏显示控制方式：空格向上翻页；b向上翻页；q退出）



② git log --pretty=oneline：简洁的显示日志信息；

```
$ git log --pretty=oneline
789fc2edf4789764be944be96e3009dcaee8b9a5 (HEAD -> master) my second commit
553501e8ffbeb6bf71bb8bff6e4061484ea7a314 test woshi yige youxiu deren!
a4f7317dac9d8a7f7c25ed90cba20f831526a3f1 first commit.new file test.txt
```

③ git log --oneline：在git log --pretty=oneline的基础上将hash值缩短一部分；

```
$ git log --oneline
789fc2e (HEAD -> master) my second commit
553501e test woshi yige youxiu deren!
a4f7317 first commit.new file test.txt
```

④git reflog：在oneline的基础上，HEADE@{移动到当前版本需要多少步}显示出从当前指针到指定的位置需要移动的步数；

```
$ git reflog
789fc2e (HEAD -> master) HEAD@{0}: commit: my second commit
553501e HEAD@{1}: commit: test woshi yige youxiu deren!
a4f7317 HEAD@{2}: commit (initial): first commit.new file test.txt
```

（5）前进后退

本质：就是移动git reflog中的HEAD指针；

①基于索引值操作（推荐）：git reset --hard 局部索引值；（其中局部索引值是在查看日志git reflog的第一列的值，不用刻意的记是前进还是后退，只要索引值是对的就可以）

```
$ git reset --hard a4f7317
HEAD is now at a4f7317 first commit.new file test.txt
```

②使用^符号：（只能后退一个^符号表示向后退一步，俩个^符号表示向后退俩步） git reset --hard HEAD^^

③使用~符号：（也只能后退）git reset --hard HEAD~2表示向后退2步，入股想要退n步，就将2变成n就可以。

（git reset 后的三个参数详解：（使用”git help 命令名称“查看对应命令的帮助文档）

①--soft参数：仅仅在本地库移动HEAD指针，使得暂存区和工作区凸显出来。

②--mixed参数：在本地库移动HEAD指针，并且会重置暂存区，使得工作区凸显出来。

③--hard参数：在本地库移动HEAD指针，重置暂存区，并且还会重置工作区。

）

（6）删除文件并召回：（任何操作只要提交到本地库中就是不可磨灭的，是指提交到版本库中的记录是删除不了的，除非将整个本地库删除。但是由于各个版本记录的存在，文件删除从工作区来讲，删除的文件恢复到它没有被删除的版本记录就可以将文件找回来）

①删除文件：”rm 文件名+文件后缀“表示在当前目录下删除指定的文件。

②删除文件之后执行了add和commit操作，可以配合使用git reflog和git reset --hard commit操作的索引值，就可以将删除的文件找回来，但是如果进入deleted操作的索引值就不能将删除的文件找回来。

找回文件的前提：删除前，文件存在时的状态已经提交到了本地库。

操作：git reset --hard 指针位置；

①删除操作已经提交到本地库，指针位置指向历史记录。

②删除操作尚未提交到本地库，指针位置使用HEAD就可以。

（7）比较文件差异：

git diff 文件名：查看修改后的文件和暂存区中文件的差异。

①git diff 文件名：将工作区中的文件和暂存区中的文件进行比较；

②git diff 本地库中历史版本 文件名：将工作区中的文件和本地库中历史记录进行比较。不指定文件名，表示比较工作区中的所有文件。

3，分支管理

进入vim编辑器，按ESC，然后输入：set nu即可以对文件进行修改。

（1）什么是分支？

在版本控制过程中，使用多条线同时推进多个任务。

（2）分支的好处：①同时并行推进多个功能开发，提高开发效率；②各个分支在开发过程中，如果某一个分支开发失败，不会对其他分支有任何影响，失败的分支重新开始即可。

（3）分支操作

①创建分支

git branch 分支名

②查看分支

git branch -v

③切换分支

git checkout 分支名

④合并分支

首先切换到接受修改的分支上（被合并，增加新内容）上。”git checkout 被合并的分支名“

然后执行merge命令即可。“git merge 有新内容的分支”

```
$ git checkout master
Switched to branch 'master'

Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/WeChat (master)
$ git branch -v
  hot_fix 9952574 test2
* master  e67ad4b test2 de dierci tijiao

Anly@DESKTOP-2API2N9 MINGW64 ~/Desktop/git_learn/WeChat (master)
$ git merge hot_fix
Updating e67ad4b..9952574
Fast-forward
 te2.txt | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
```

⑤解决冲突

在俩个分支中同时修改同一个文件中的同一行记录，并且都提交到本地库中，若此时执行合并操作的话，会报以下的错误。

```
$ git merge hot_fix
Auto-merging te2.txt
CONFLICT (content): Merge conflict in te2.txt
Automatic merge failed; fix conflicts and then commit the result.
```

a、冲突的表现：6、7、8行表示当前分支的内容；9、10行表示另一个分支的内容。

```
 4 ddd
 5 dd
 6 <<<<<< HEAD
 7 master
 8 =======
 9 hot_fix
10 >>>>>>> hot_fix
11
12
```

经商量决定后将多余的符号删除，达到满意的效果然后退出提交。

b、冲突的解决：

第一步：编辑文件，删除特殊符号；

第二步：把文件修改到满意程度，保存退出；

第三步：git add 文件名；

第四步：git commit -m “日志信息”。（注意：commit一定不能带具体的文件名）