THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.
滚动浏览详情

该文档基于MIT课程

在git中，文件、文件夹、commit都是怎么组成的？

文件(blob)

typeof(blob) = array -> **一个文件是bit组成的数组**

文件夹(tree)

typeof(tree) = map<string, tree | blob> -> 文件夹实际上是一个字典/哈希图；string代表文件名，tree|blob代表这个名下的文件/文件夹的内容

commit

typeof(commit) = struct{
parents = array # 代表该commit的历史记录,数组形式存储
author,message等meta data
snapshot = tree
}

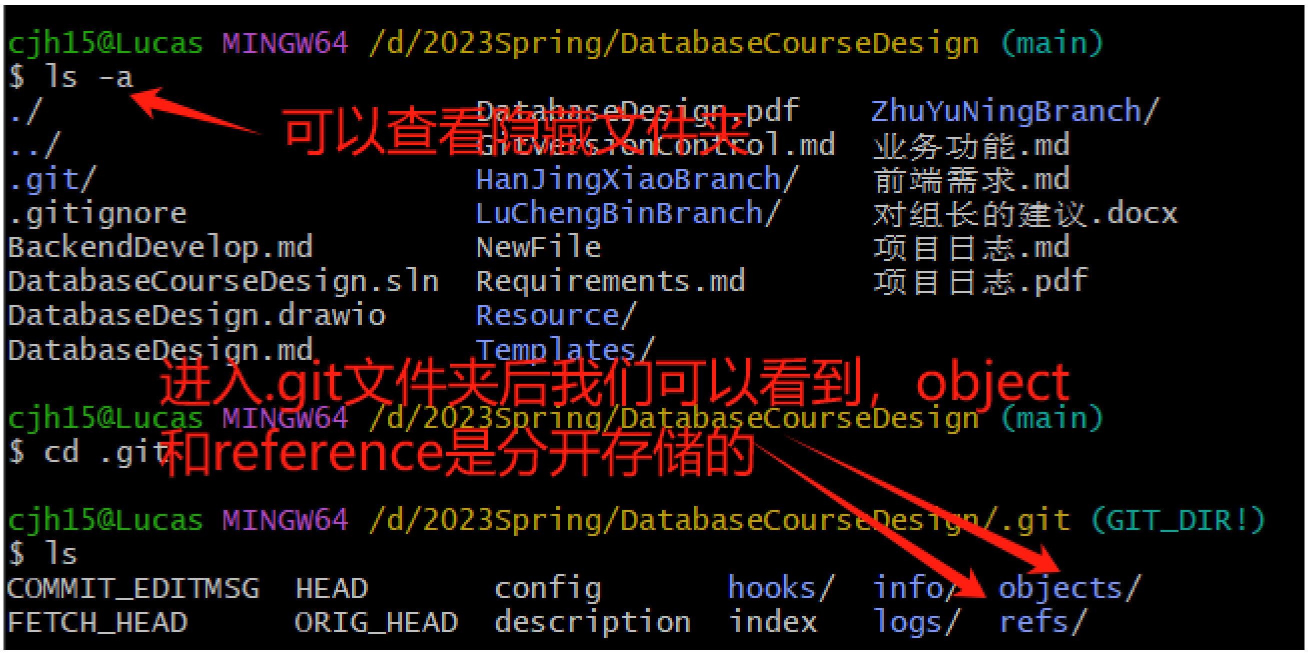
git是如何处理这三种数据结构的

git**平等对待上述的数据结构**

type object = blob | tree | commit -> 数据类型object就是“平等”的体现
ObjectStorage = map<string,object>
如果把文件/文件夹/commit提交到ObjectStorage，**实际上是对字典/哈希表进行添加操作**，写入到硬盘里；取文件就跟从哈希表里取是一样的

reference = map<string,string> -> reference实现的是**人能看得懂的**文件名到**16进制的文件哈希值**的映射关系

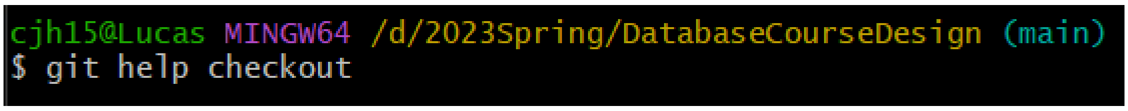
git仓库的组成


```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ ls -a
./                          DatabaseDesign.pdf        ZhuYuNingBranch/
../                         GitVersionControl.md      业务功能.md
.git/                       HanJingXiaoBranch/        前端需求.md
.gitignore                  LuChengBinBranch/         对组长的建议.docx
BackendDevelop.md           NewFile                   项目日志.md
DatabaseCourseDesign.sln    Requirements.md           项目日志.pdf
DatabaseDesign.drawio       Resource/
DatabaseDesign.md           Templates/

cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ cd .git

cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign/.git (GIT_DIR!)
$ ls
COMMIT_EDITMSG  HEAD        config       hooks/  info/  objects/
FETCH_HEAD      ORIG_HEAD   description  index   logs/  refs/
```

git指令

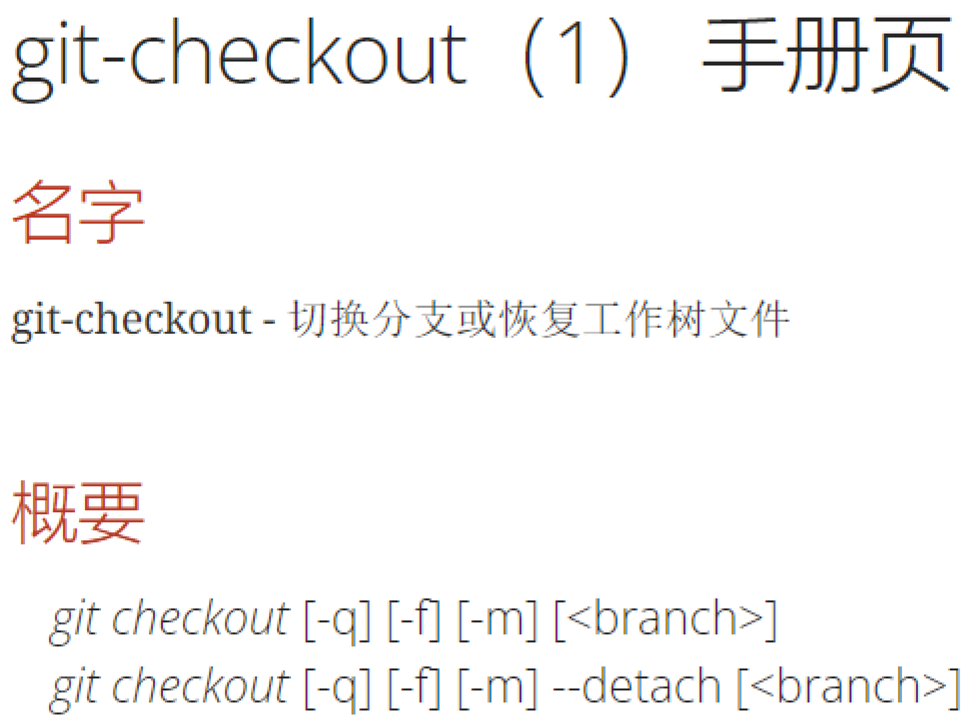
`git help`

输入`git help <git-command>`(""代表对哪个git指令进行提问)
图片中对`checkout`指令进行提问



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git help checkout
```

输入后会打开一个操作指南窗口



git-checkout（1） 手册页

名字

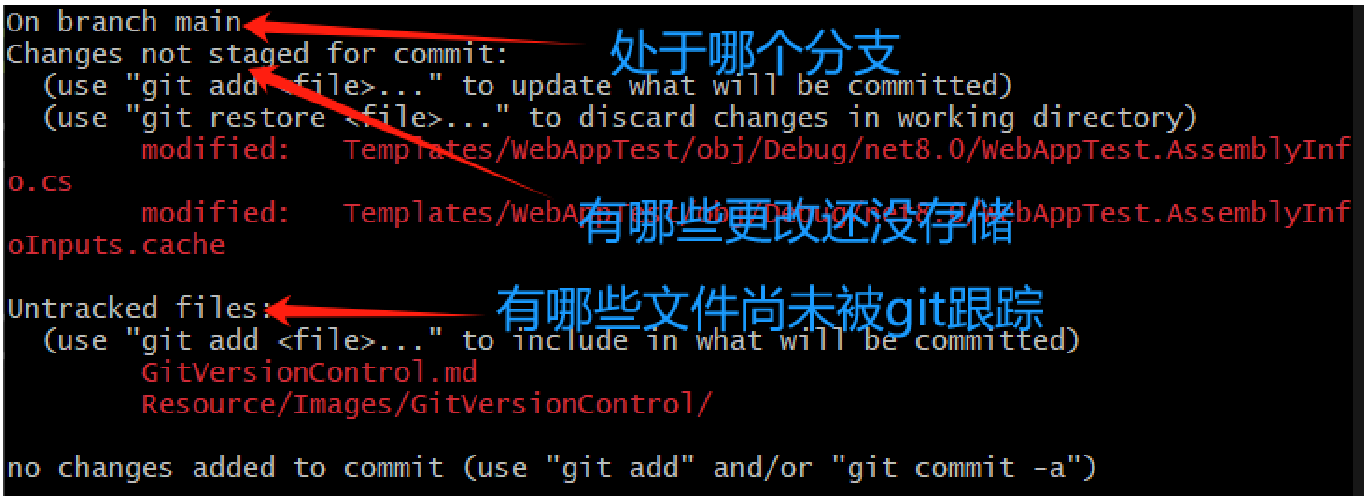
git-checkout - 切换分支或恢复工作树文件

概要

git checkout [-q] [-f] [-m] [<branch>]
git checkout [-q] [-f] [-m] --detach [<branch>]

`git status`

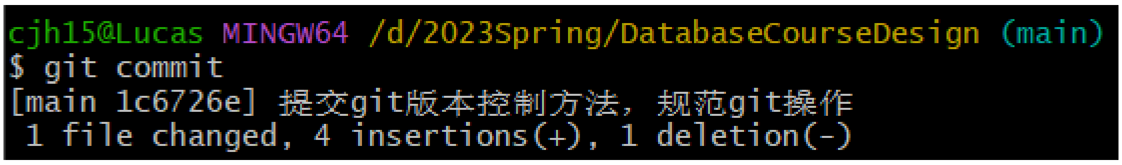
该指令可以查询当前我的git状态



```
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Templates/WebAppTest/obj/Debug/net8.0/WebAppTest.AssemblyInf
o.cs
        modified:   Templates/WebAppTest/obj/Debug/net8.0/WebAppTest.AssemblyInf
oInputs.cache

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        GitVersionControl.md
        Resource/Images/GitVersionControl/

no changes added to commit (use "git add" and/or "git commit -a")
```

`git commit`

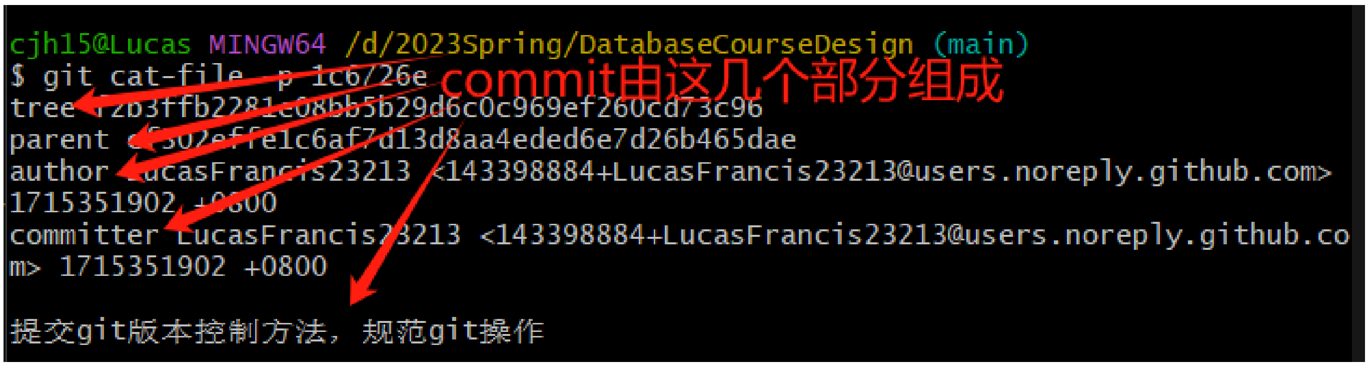
该指令会创建一次提交



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git commit
[main 1c6726e] 提交git版本控制方法，规范git操作
 1 file changed, 4 insertions(+), 1 deletion(-)
```

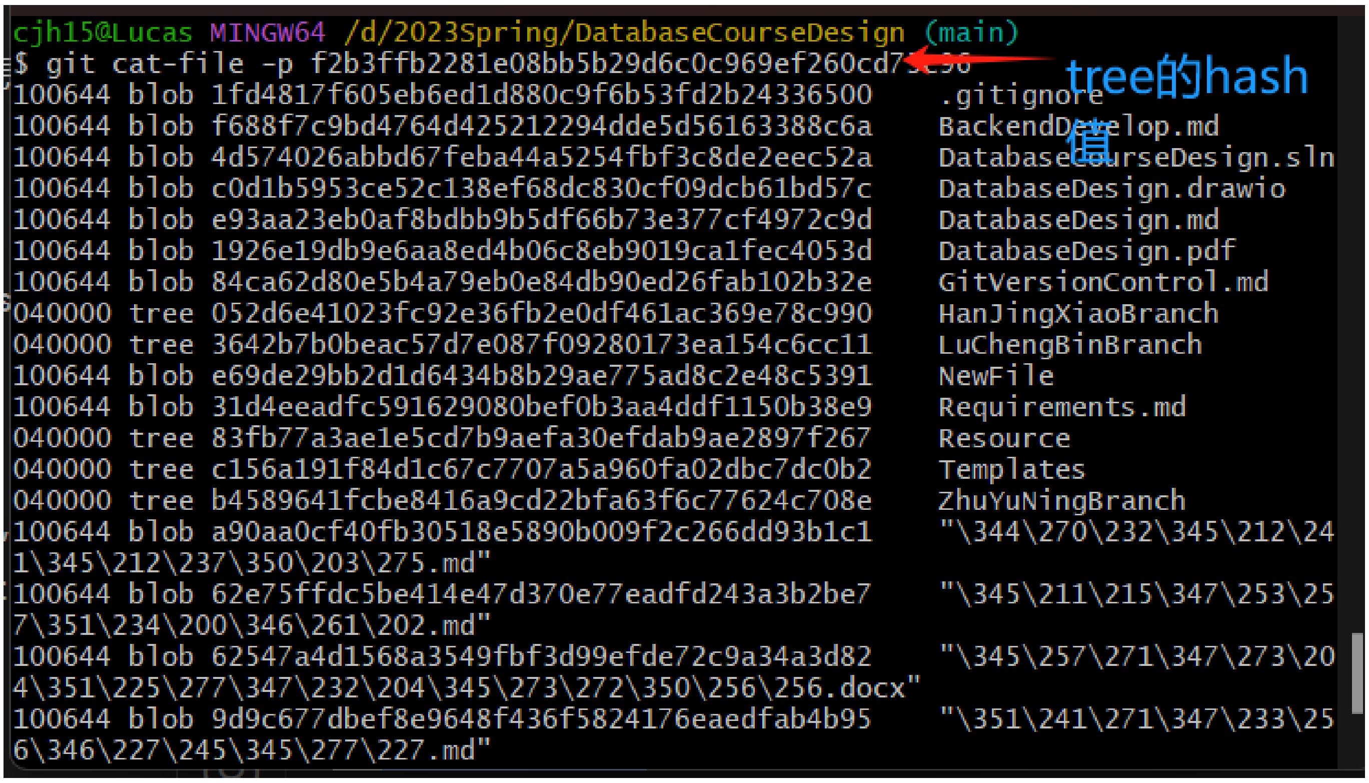
只使用`git commit`后会跳转到vim文本编辑器创建commit信息，如果用不习惯vim可以使用指令`git config --global core.editor "nano"`换成nano编辑器

如果使用`git cat-file -p <commit-hash>`，我们就能看到这个commit的**完整的提交信息**

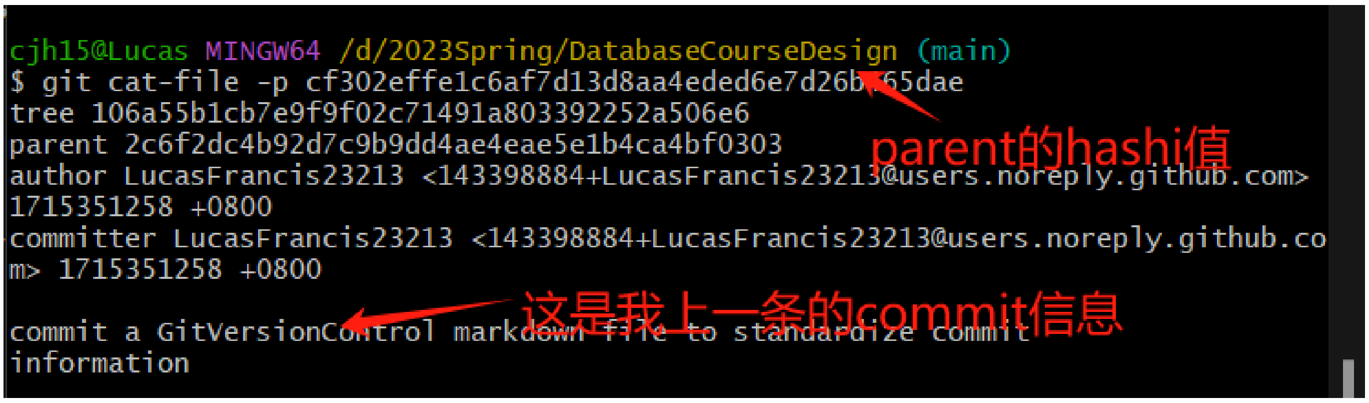


```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git cat-file -p 1c6726e
tree f2b3ffb2281e08bb5b29d6c0c969ef260cd73c96
parent cf302effe1c6af7d13d8aa4eded6e7d26b465dae
author LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.com>
1715351902 +0800
committer LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m> 1715351902 +0800

提交git版本控制方法，规范git操作
```

同理，也可以通过`git cat-file -p <tree-hash>/<parent-hash>`查看该commit的**tree是谁**以及这个commit的**历史提交**



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git cat-file -p f2b3ffb2281e08bb5b29d6c0c969ef260cd73c96
100644 blob 1fd4817f605eb6ed1d880c9f6b53fd2b24336500    .gitignore
100644 blob f688f7c9bd4764d425212294dde5d56163388c6a    BackendDevelop.md
100644 blob 4d574026abbd67feba44a5254fbf3c8de2eec52a    DatabaseCourseDesign.sln
100644 blob c0d1b5953ce52c138ef68dc830cf09dcb61bd57c    DatabaseDesign.drawio
100644 blob e93aa23eb0af8bdbb9b5df66b73e377cf4972c9d    DatabaseDesign.md
100644 blob 1926e19db9e6aa8ed4b06c8eb9019ca1fec4053d    DatabaseDesign.pdf
100644 blob 84ca62d80e5b4a79eb0e84db90ed26fab102b32e    GitVersionControl.md
040000 tree 052d6e41023fc92e36fb2e0df461ac369e78c990    HanJingXiaoBranch
040000 tree 3642b7b0beac57d7e087f09280173ea154c6cc11    LuChengBinBranch
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    NewFile
100644 blob 31d4eeadfc591629080bef0b3aa4ddf1150b38e9    Requirements.md
040000 tree 83fb77a3ae1e5cd7b9aefa30efdab9ae2897f267    Resource
040000 tree c156a191f84d1c67c7707a5a960fa02dbc7dc0b2    Templates
040000 tree b4589641fcbe8416a9cd22bfa63f6c77624c708e    ZhuYuNingBranch
100644 blob a90aa0cf40fb30518e5890b009f2c266dd93b1c1    "\344\270\232\345\212\24
1\345\212\237\350\203\275.md"
100644 blob 62e75ffdc5be414e47d370e77eadfd243a3b2be7    "\345\211\215\347\253\25
7\351\234\200\346\261\202.md"
100644 blob 62547a4d1568a3549fbf3d99efde72c9a34a3d82    "\345\257\271\347\273\20
4\351\225\277\347\232\204\345\273\272\350\256\256.docx"
100644 blob 9d9c677dbef8e9648f436f5824176eaedfab4b95    "\351\241\271\347\233\25
6\346\227\245\345\277\227.md"
```

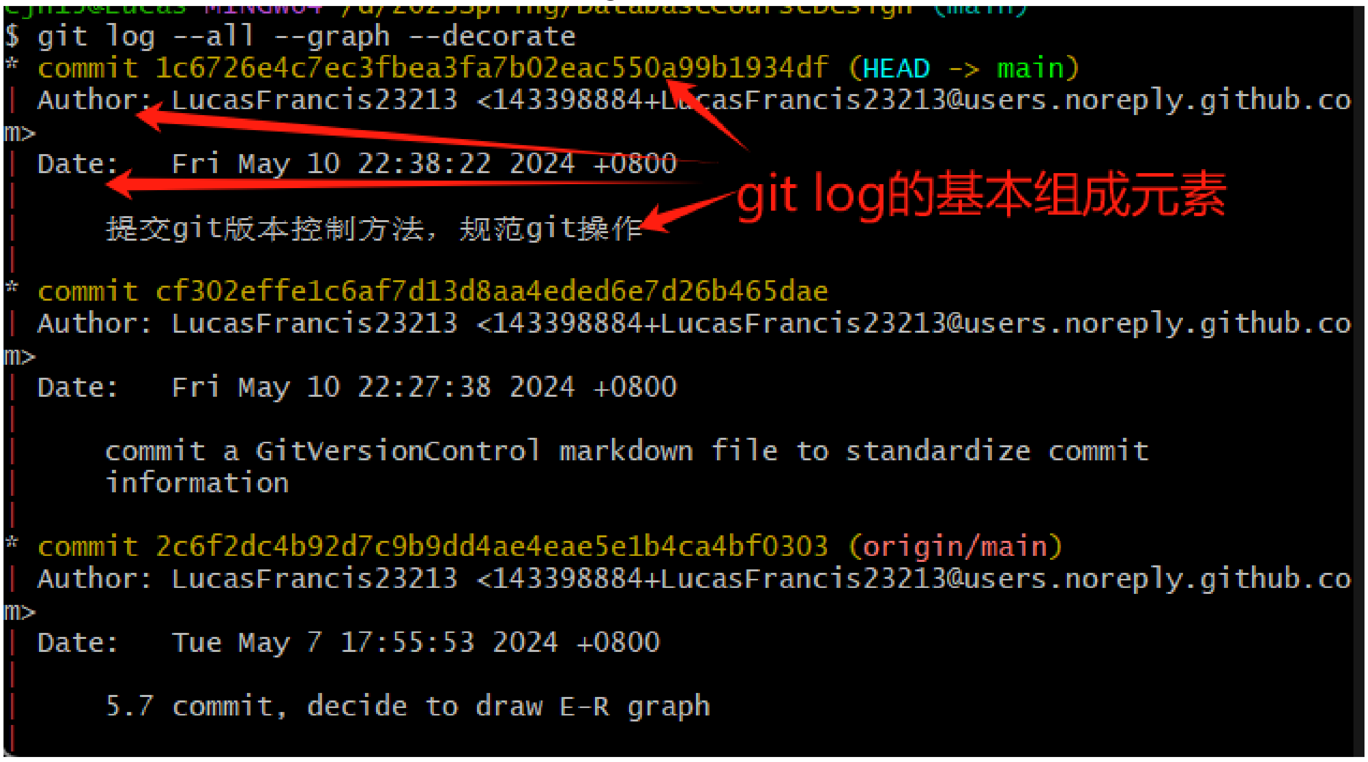


```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git cat-file -p cf302effe1c6af7d13d8aa4eded6e7d26b465dae
tree 106a55b1cb7e9f9f02c71491a803392252a506e6
parent 2c6f2dc4b92d7c9b9dd4ae4eae5e1b4ca4bf0303
author LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.com>
1715351258 +0800
committer LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m> 1715351258 +0800

commit a GitVersionControl markdown file to standardize commit
information
```

git log

该指令可以查看提交的记录

`git log --all --graph --decorate`可以将log按时间先后呈现出来



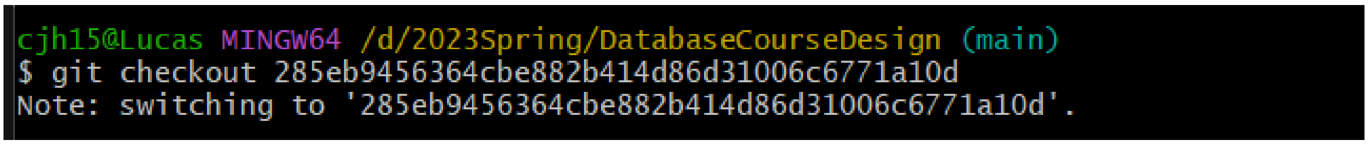
```
$ git log --all --graph --decorate
* commit 1c6726e4c7ec3fbea3fa7b02eac550a99b1934df (HEAD -> main)
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Fri May 10 22:38:22 2024 +0800
|
|     提交git版本控制方法，规范git操作
|
* commit cf302effe1c6af7d13d8aa4eded6e7d26b465dae
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Fri May 10 22:27:38 2024 +0800
|
|     commit a GitVersionControl markdown file to standardize commit
|     information
|
* commit 2c6f2dc4b92d7c9b9dd4ae4eae5e1b4ca4bf0303 (origin/main)
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Tue May 7 17:55:53 2024 +0800
|
|     5.7 commit, decide to draw E-R graph
|
```

输入wq退出(跟退出vim编辑器是一样的)

git checkout

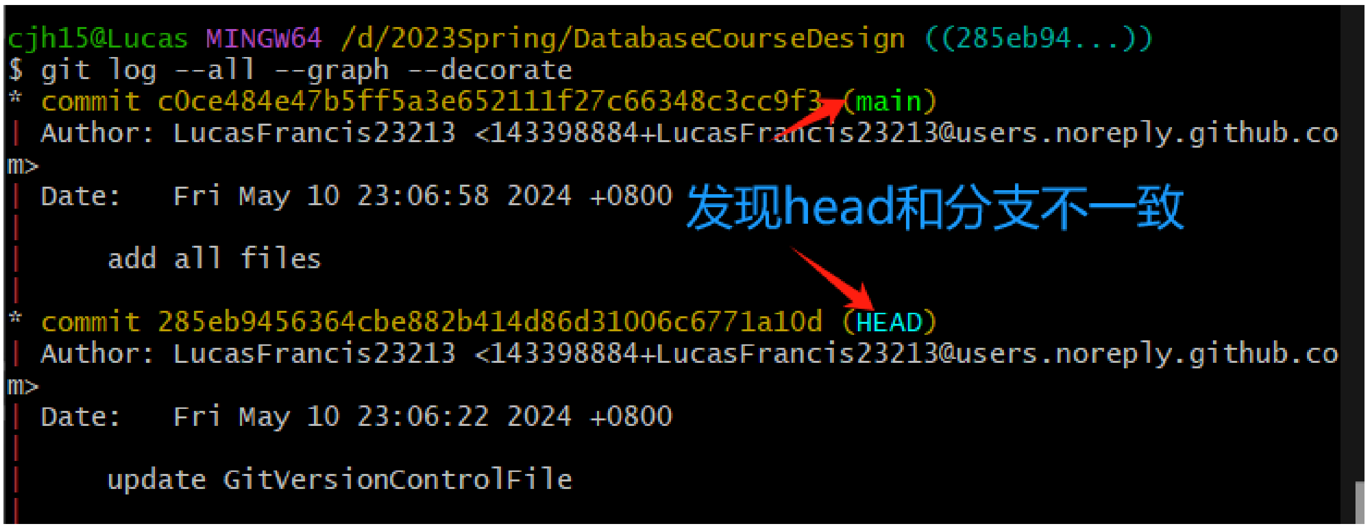
该指令可以让你退回到某个commit处

退回到指定hash值的commit处



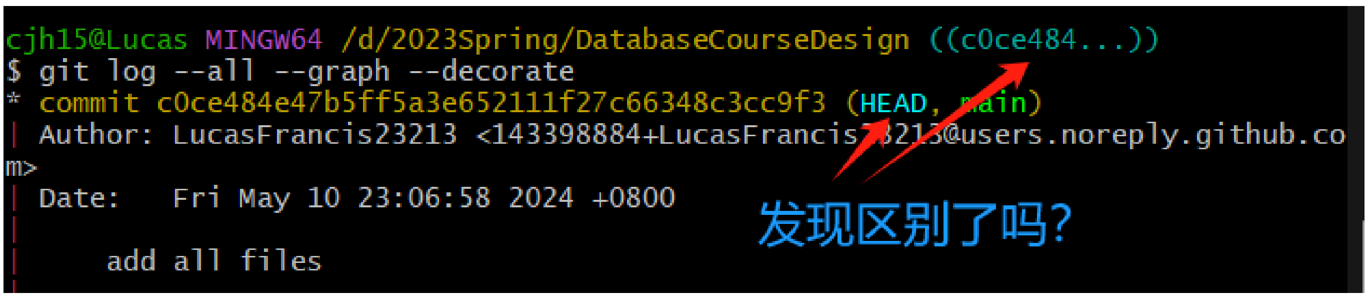
```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git checkout 285eb9456364cbe882b414d86d31006c6771a10d
Note: switching to '285eb9456364cbe882b414d86d31006c6771a10d'.
```

`git checkout`和`git log`是**相互独立的**

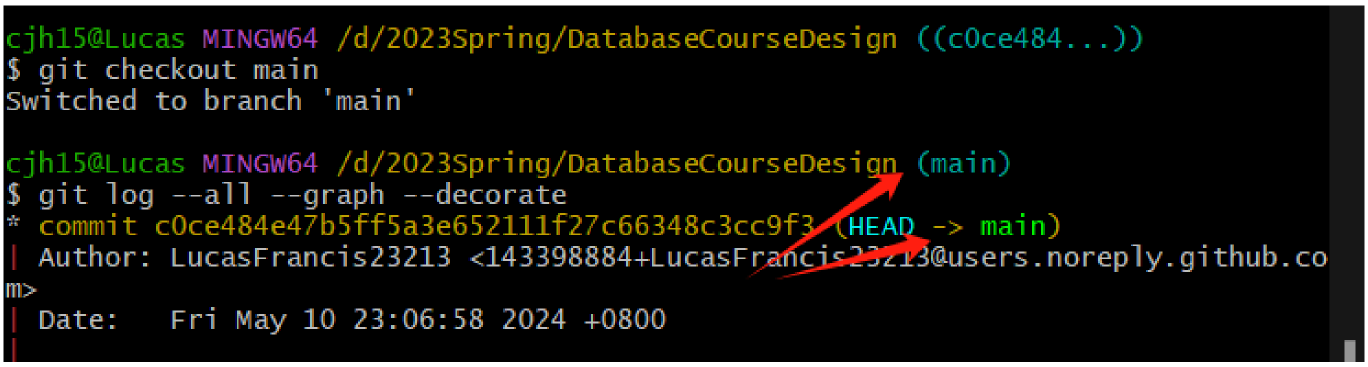


```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign ((285eb94...))
$ git log --all --graph --decorate
* commit c0ce484e47b5ff5a3e652111f27c66348c3cc9f3 (main)
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Fri May 10 23:06:58 2024 +0800
|
|     add all files
|
* commit 285eb9456364cbe882b414d86d31006c6771a10d (HEAD)
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Fri May 10 23:06:22 2024 +0800
|
|     update GitVersionControlFile
|
```

观察：`git checkout <commit-hash>` 和 `git checkout <branch name>` 的区别



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign ((c0ce484...))
$ git log --all --graph --decorate
* commit c0ce484e47b5ff5a3e652111f27c66348c3cc9f3 (HEAD, main)
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Fri May 10 23:06:58 2024 +0800
|
|     add all files
```



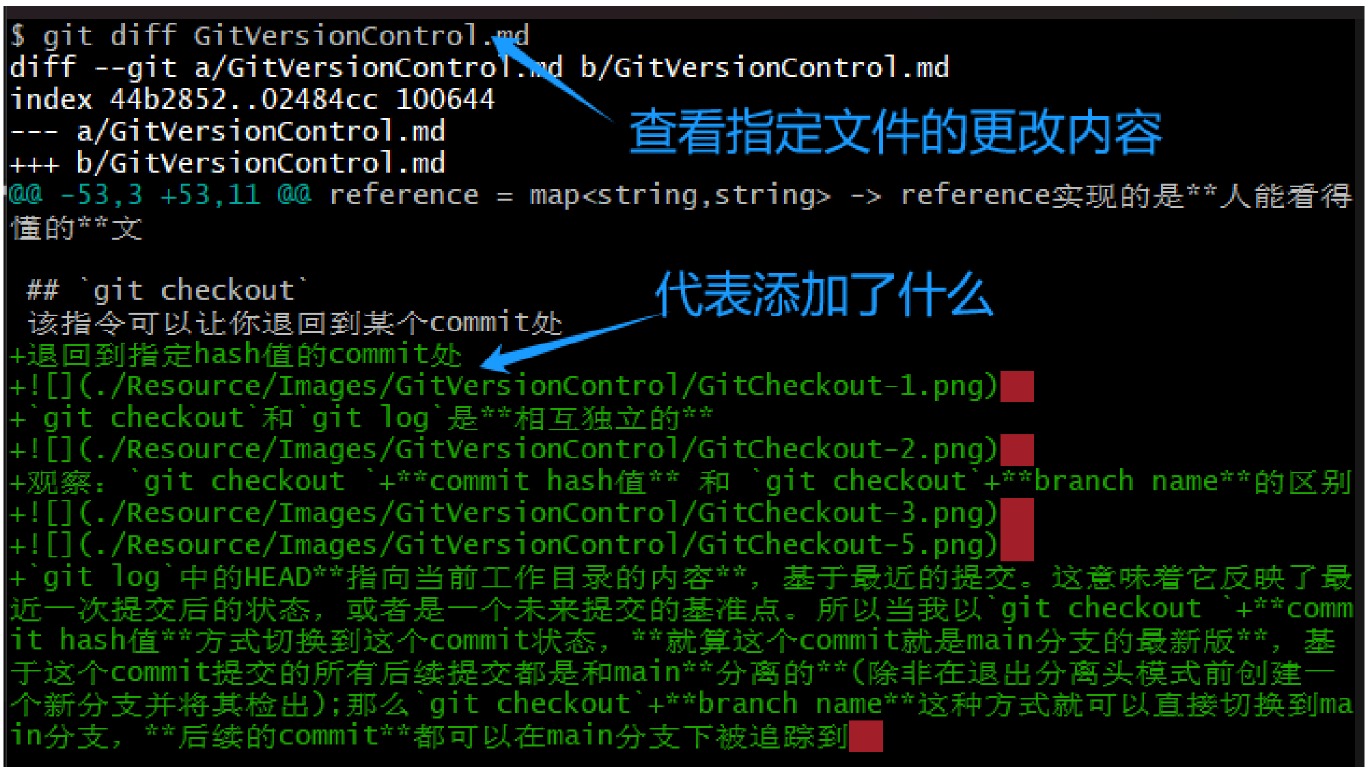
```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign ((c0ce484...))
$ git checkout main
Switched to branch 'main'

cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git log --all --graph --decorate
* commit c0ce484e47b5ff5a3e652111f27c66348c3cc9f3 (HEAD -> main)
| Author: LucasFrancis23213 <143398884+LucasFrancis23213@users.noreply.github.co
m>
| Date:   Fri May 10 23:06:58 2024 +0800
```

`git log`中的HEAD**指向当前工作目录的内容**，基于最近的提交。这意味着它反映了最近一次提交后的状态，或者是一个未来提交的基准点。所以当我以`git checkout <commit-hash>`方式切换到这个commit状态，**就算这个commit就是main分支的最新版**，基于这个commit提交的所有后续提交都是和main**分离的**(除非在退出分离头模式前创建一个新分支并将其检出);那么`git checkout <branch name>`这种方式就可以直接切换到main分支，**后续的commit**都可以在main分支下被追踪到

当然切换分支也可以使用`git switch <branch-name>`实际上使用这个指令切换分支更安全

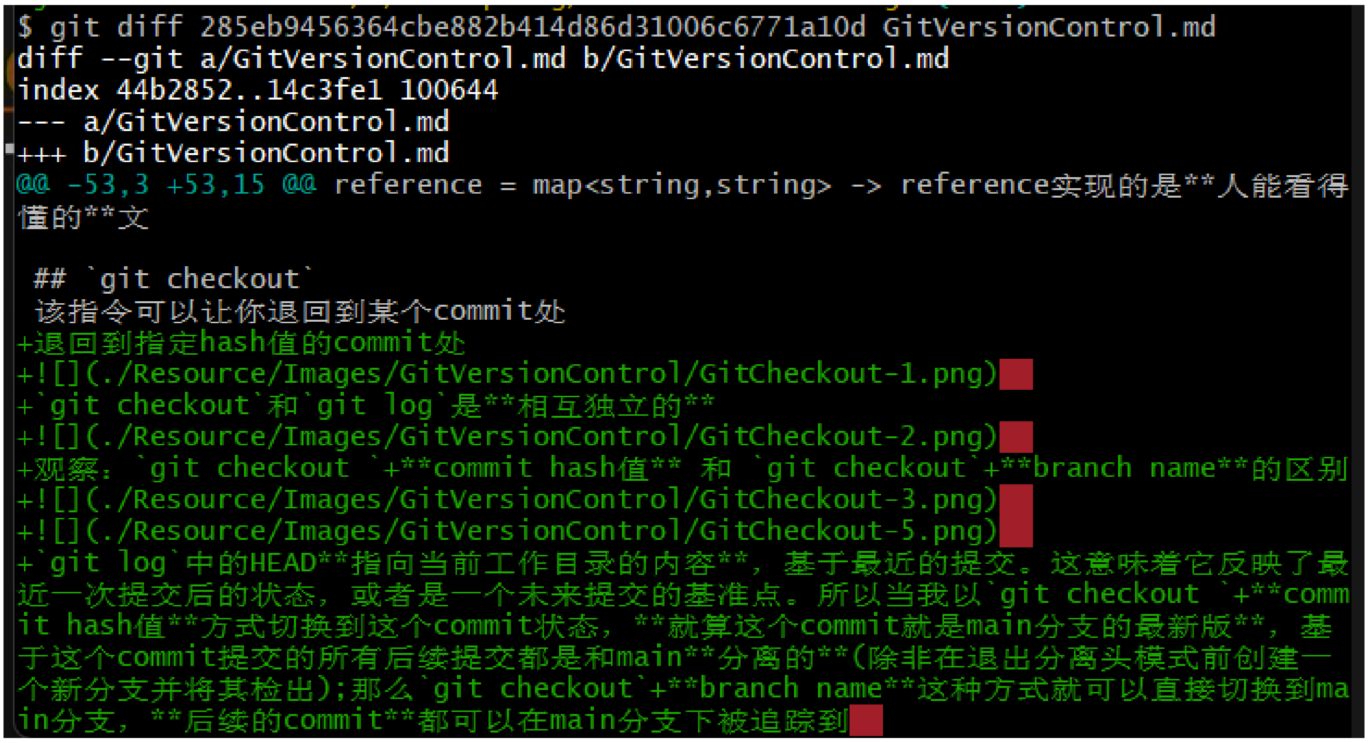
`git diff`

该指令可以查看两次提交中到底改变了什么内容
缺省情况下，`git diff`比较的是**当前工作区**和**HEAD指向的commit**的内容差异



```
$ git diff GitVersionControl.md
diff --git a/GitVersionControl.md b/GitVersionControl.md
index 44b2852..02484cc 100644
--- a/GitVersionControl.md
+++ b/GitVersionControl.md
@@ -53,3 +53,11 @@ reference = map<string,string> -> reference实现的是**人能看得
懂的**文

 ## `git checkout`
 该指令可以让你退回到某个commit处
+退回到指定hash值的commit处
+![](./Resource/Images/GitVersionControl/GitCheckout-1.png)
+`git checkout`和`git log`是**相互独立的**
+![](./Resource/Images/GitVersionControl/GitCheckout-2.png)
+观察: `git checkout `+**commit hash值** 和 `git checkout`+**branch name**的区别
+![](./Resource/Images/GitVersionControl/GitCheckout-3.png)
+![](./Resource/Images/GitVersionControl/GitCheckout-5.png)
+`git log`中的HEAD**指向当前工作目录的内容**, 基于最近的提交。这意味着它反映了最
近一次提交后的状态, 或者是一个未来提交的基准点。所以当我以`git checkout `+**comm
it hash值**方式切换到这个commit状态, **就算这个commit就是main分支的最新版**, 基
于这个commit提交的所有后续提交都是和main**分离的**(除非在退出分离头模式前创建一
个新分支并将其检出);那么`git checkout`+**branch name**这种方式就可以直接切换到ma
in分支, **后续的commit**都可以在main分支下被追踪到
```

如果要查看**某次commit提交后某个文件改了什么**，可以使用`git diff <commit-hash> <file-name>`，这就相当于把HEAD改为了某次提交

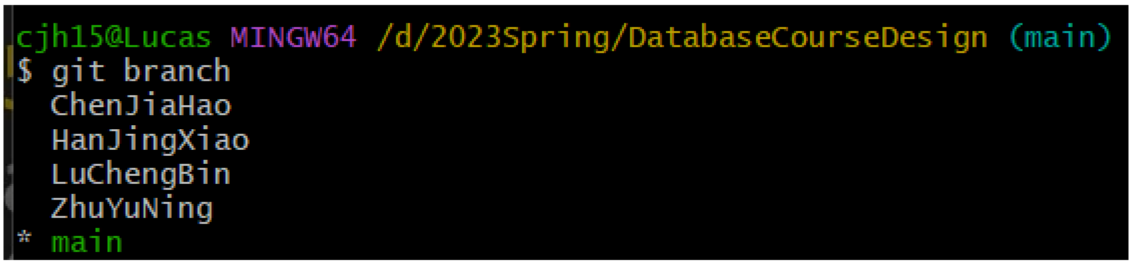


```
$ git diff 285eb9456364cbe882b414d86d31006c6771a10d GitVersionControl.md
diff --git a/GitVersionControl.md b/GitVersionControl.md
index 44b2852..14c3fe1 100644
--- a/GitVersionControl.md
+++ b/GitVersionControl.md
@@ -53,3 +53,15 @@ reference = map<string,string> -> reference实现的是**人能看得懂的**文

 ## `git checkout`
 该指令可以让你退回到某个commit处
+退回到指定hash值的commit处
+![](./Resource/Images/GitVersionControl/GitCheckout-1.png)
+`git checkout`和`git log`是**相互独立的**
+![](./Resource/Images/GitVersionControl/GitCheckout-2.png)
+观察：`git checkout `+**commit hash值** 和 `git checkout`+**branch name**的区别
+![](./Resource/Images/GitVersionControl/GitCheckout-3.png)
+![](./Resource/Images/GitVersionControl/GitCheckout-5.png)
+`git log`中的HEAD**指向当前工作目录的内容**，基于最近的提交。这意味着它反映了最近一次提交后的状态，或者是一个未来提交的基准点。所以当我以`git checkout `+**commit hash值**方式切换到这个commit状态，**就算这个commit就是main分支的最新版**，基于这个commit提交的所有后续提交都是和main**分离的**(除非在退出分离头模式前创建一个新分支并将其检出);那么`git checkout`+**branch name**这种方式就可以直接切换到main分支，**后续的commit**都可以在main分支下被追踪到
```

`git diff`实际上可以接受两个参数，`git diff <commit-hash-1> <commit-hash-2> <file-name>`，这个指令代表从`<commit-hash-1>`到`<commit-hash-2>`之间`<file-name>`这个文件做了哪些改变

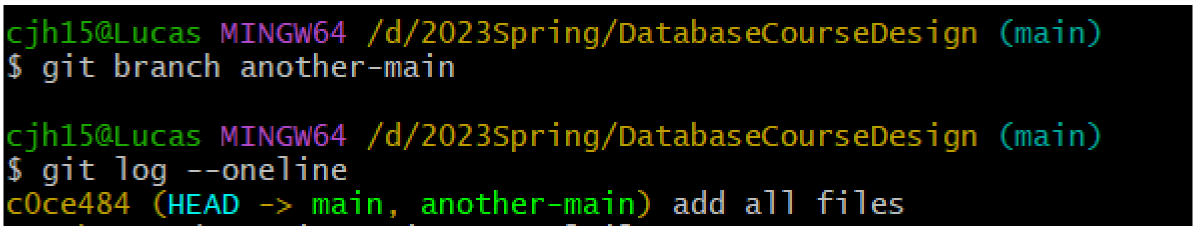
`git branch`

该指令会列出本地的所有分支

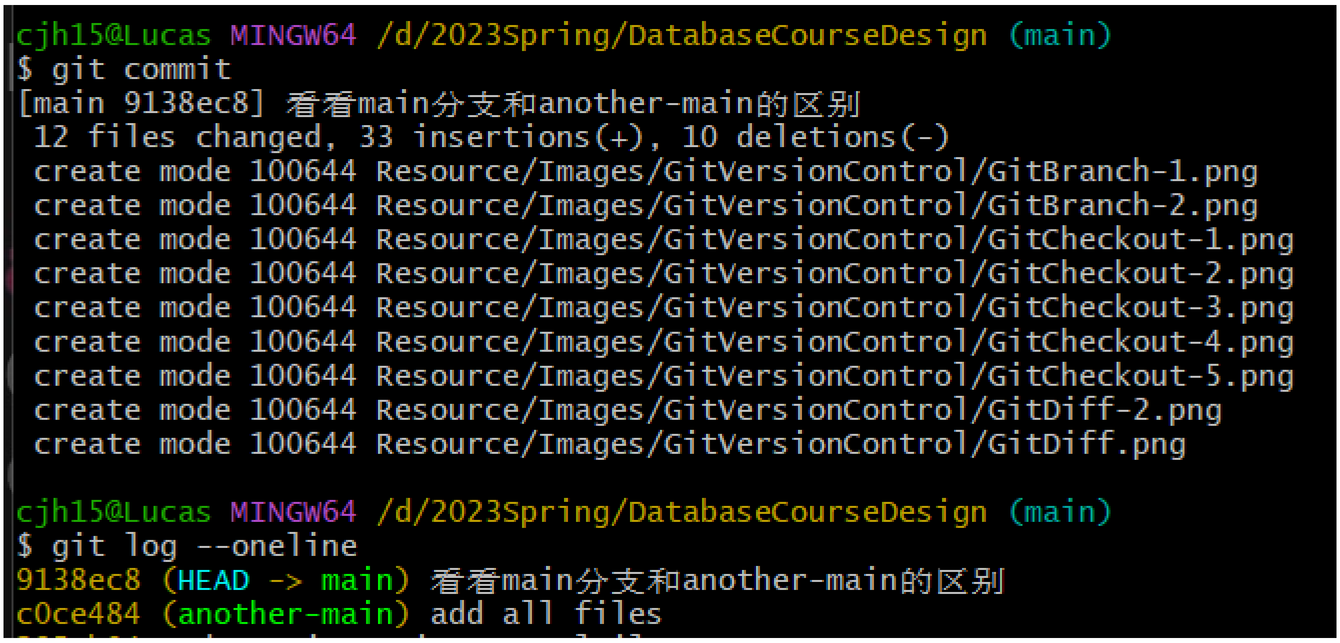


```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git branch
  ChenJiaHao
  HanJingXiao
  LuChengBin
  ZhuYuNing
* main
```

`git branch <branch-name>`指令会基于当前HEAD所指的内容创建一个新的平行分支



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git branch another-main

cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git log --oneline
c0ce484 (HEAD -> main, another-main) add all files
```



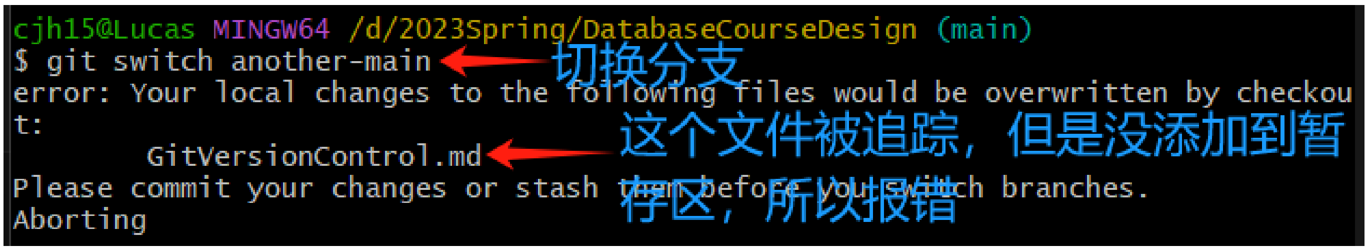
```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git commit
[main 9138ec8] 看看main分支和another-main的区别
 12 files changed, 33 insertions(+), 10 deletions(-)
 create mode 100644 Resource/Images/GitVersionControl/GitBranch-1.png
 create mode 100644 Resource/Images/GitVersionControl/GitBranch-2.png
 create mode 100644 Resource/Images/GitVersionControl/GitCheckout-1.png
 create mode 100644 Resource/Images/GitVersionControl/GitCheckout-2.png
 create mode 100644 Resource/Images/GitVersionControl/GitCheckout-3.png
 create mode 100644 Resource/Images/GitVersionControl/GitCheckout-4.png
 create mode 100644 Resource/Images/GitVersionControl/GitCheckout-5.png
 create mode 100644 Resource/Images/GitVersionControl/GitDiff-2.png
 create mode 100644 Resource/Images/GitVersionControl/GitDiff.png

cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git log --oneline
9138ec8 (HEAD -> main) 看看main分支和another-main的区别
c0ce484 (another-main) add all files
```

`git branch --set-upstream-to=<remote-repo-name>/<remote-branch>`，该指令可以设定push时缺省状态下会提交到哪个远程分支

`git merge`

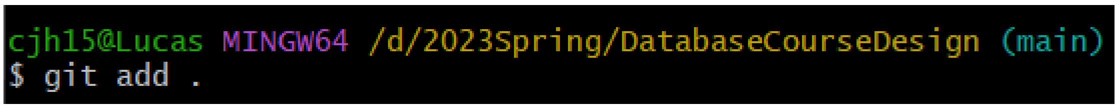
该指令可以视为`git branch`的反操作，`git branch`是创建分支的，`git merge`是合并分支的

在**切换分支/合并分支**前，如果没有将本分支**正在被追踪(tracked)文件添加到暂存区(staging area)**，会报错



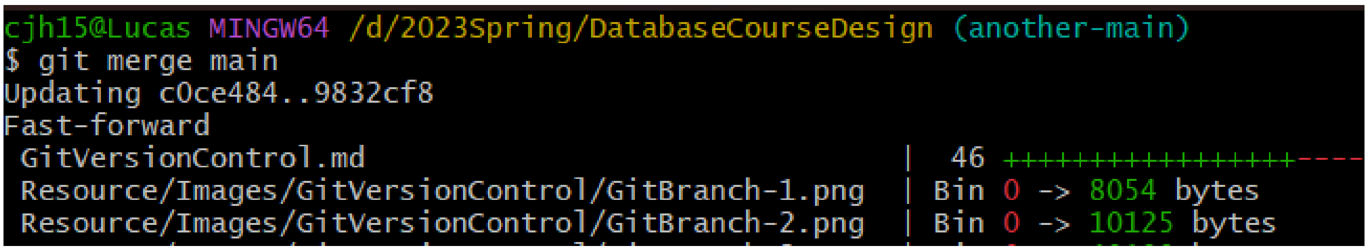
```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git switch another-main
error: Your local changes to the following files would be overwritten by checkou
t:
        GitVersionControl.md
Please commit your changes or stash them before you switch branches.
Aborting
```

使用`git add <file-name>`将文件添加到**暂存区**



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git add .
```

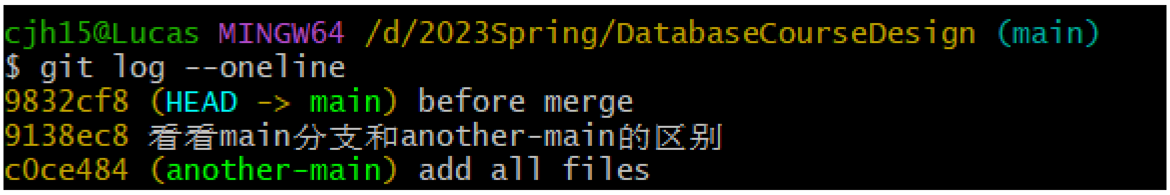
现在合并就没问题了



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (another-main)
$ git merge main
Updating c0ce484..9832cf8
Fast-forward
 GitVersionControl.md                              |  46 +++++++++++++++++++----
 Resource/Images/GitVersionControl/GitBranch-1.png | Bin 0 -> 8054 bytes
 Resource/Images/GitVersionControl/GitBranch-2.png | Bin 0 -> 10125 bytes
```

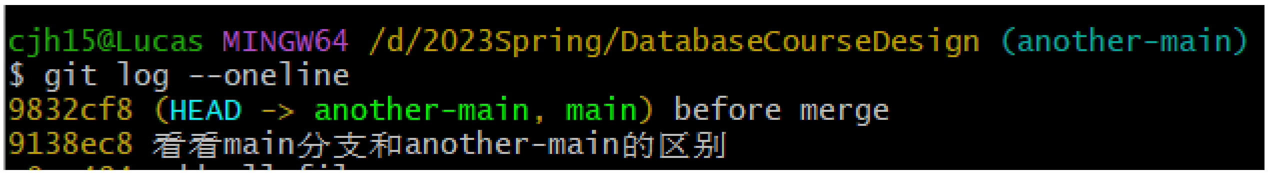
如果我所处的分支和我要合并的分支存在**父子关系**，那么合并流程就是**移动HEAD指针到合并的分支处**

merge前两分支的状态：



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git log --oneline
9832cf8 (HEAD -> main) before merge
9138ec8 看看main分支和another-main的区别
c0ce484 (another-main) add all files
```

merge后两分支的状态，**观察HEAD的变化：**

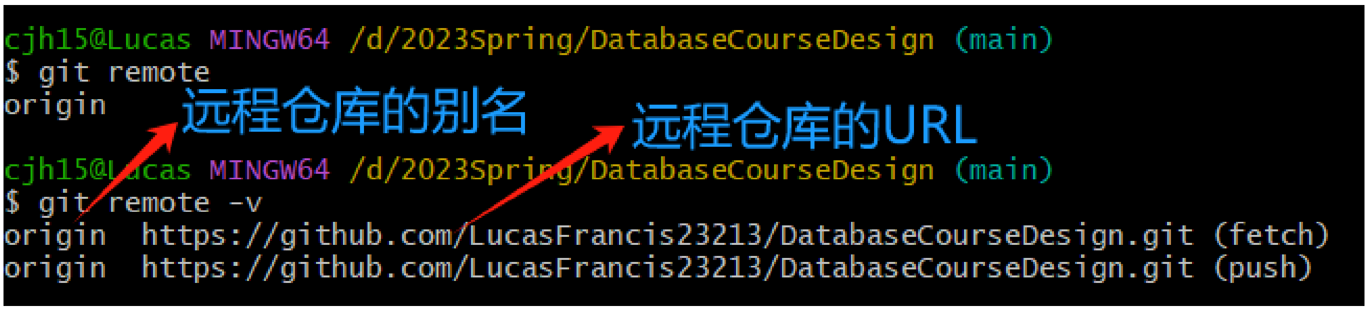


```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (another-main)
$ git log --oneline
9832cf8 (HEAD -> another-main, main) before merge
9138ec8 看看main分支和another-main的区别
```

如果存在合并冲突(merge conflicts)，VSCODE/VISUAL STUDIO会有“合并冲突管理器”的东西专门用于合并冲突

`git remote`

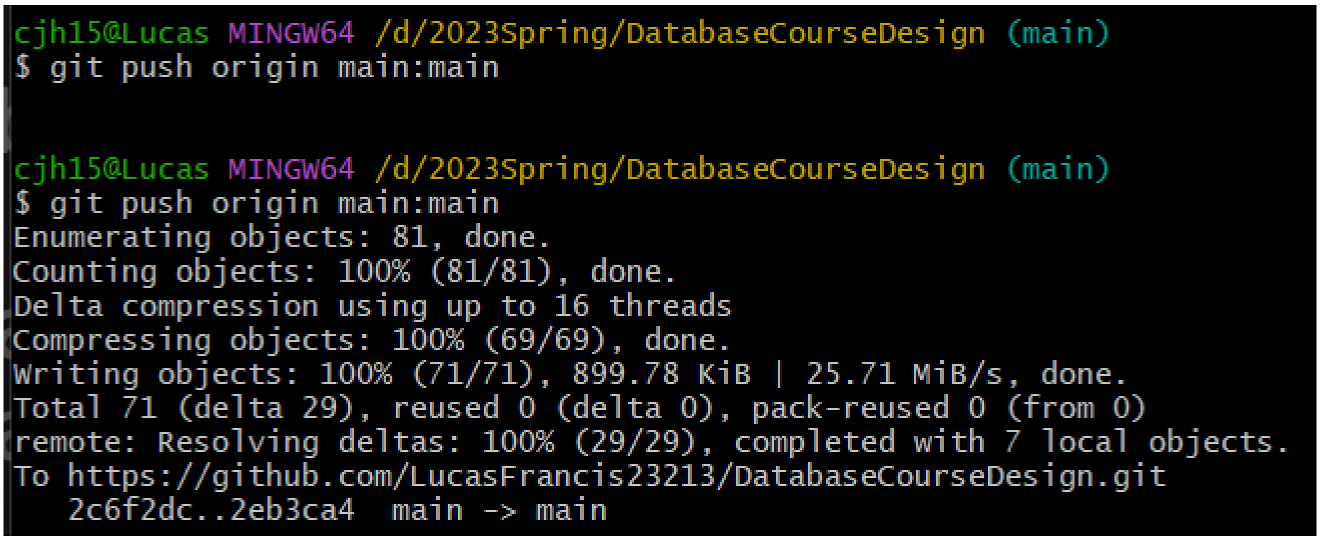
该指令可以添加、查看、删除远程仓库 `git remote add <human-readable-name> <repository-url>`指令可以为当前本地git仓库添加远程仓库,*远程url也可以是本机的地址*
`git remote -v`指令可以查看当前仓库的远程信息



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git remote
origin

cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git remote -v
origin  https://github.com/LucasFrancis23213/DatabaseCourseDesign.git (fetch)
origin  https://github.com/LucasFrancis23213/DatabaseCourseDesign.git (push)
```

`git push`

该指令可以将本地代码推送到远程仓库 `git push <remote-repo-name> <local-branch>:<remote-branch>`



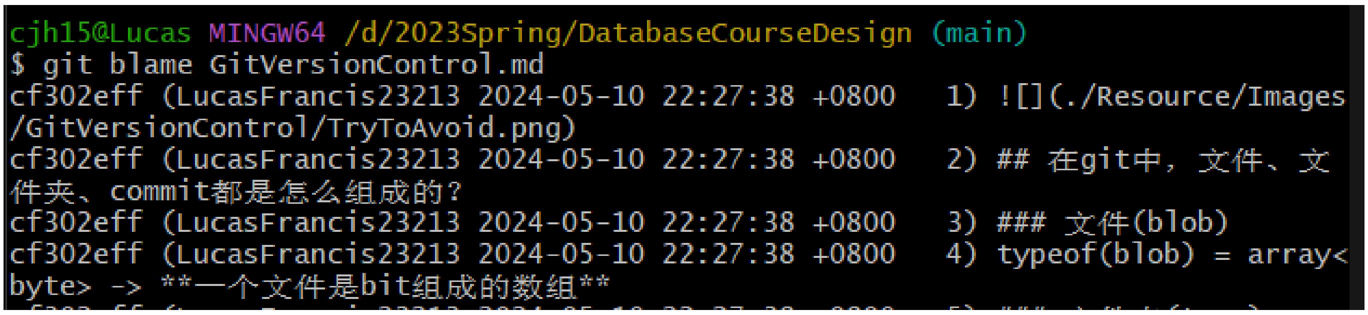
```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git push origin main:main


cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git push origin main:main
Enumerating objects: 81, done.
Counting objects: 100% (81/81), done.
Delta compression using up to 16 threads
Compressing objects: 100% (69/69), done.
Writing objects: 100% (71/71), 899.78 KiB | 25.71 MiB/s, done.
Total 71 (delta 29), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (29/29), completed with 7 local objects.
To https://github.com/LucasFrancis23213/DatabaseCourseDesign.git
   2c6f2dc..2eb3ca4  main -> main
```

`git clone`

`git clone <remote-url> <local-folder>`，该指令可以将该url链接的文件克隆到指定的本地文件夹
`git clone --shallow`这个指令**只会拉取最新版的提交**，上面的则会把**所有的版本都拉取下来**

`git blame`

`git blame <file-name>`该指令可以查看谁在什么时候改了这个文件，以及更改的具体内容



```
cjh15@Lucas MINGW64 /d/2023Spring/DatabaseCourseDesign (main)
$ git blame GitVersionControl.md
cf302eff (LucasFrancis23213 2024-05-10 22:27:38 +0800   1) ![](./Resource/Images
/GitVersionControl/TryToAvoid.png)
cf302eff (LucasFrancis23213 2024-05-10 22:27:38 +0800   2) ## 在git中, 文件、文
件夹、commit都是怎么组成的?
cf302eff (LucasFrancis23213 2024-05-10 22:27:38 +0800   3) ### 文件(blob)
cf302eff (LucasFrancis23213 2024-05-10 22:27:38 +0800   4) typeof(blob) = array<
byte> -> **一个文件是bit组成的数组**
```

上面只是我们这个项目应该会用到的git指令

每次开完会后的工作流

- 开完会之后组长会把会议记录和相关资料更新到`main`分支上
- 各位组员**先拉取**`main`分支，看这周的任务是什么、具体有什么标准和要求、什么时候ddl
- 具体做任务前，**先看说明文档/会议记录里的要求**
- **在此基础上**，如果不清楚做什么/认为文档没写清楚直接私信组长/大群里at他就行