哈尔滨工业大学

**计算机科学与技术学院/国家示范性软件学院**

**2020年秋季学期**

**《软件过程与工具》课程**

**实验报告**

**Lab 2：配置管理工具Git实践**

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **联系方式** |
| 郭茁宁 | 1183710109 | gzn00417@foxmail.com |

目 录

[1 实验要求 1](#_Toc55674453)

[2 安装Git 1](#_Toc55674454)

[2.1 本地机器上安装Git 1](#_Toc55674455)

[2.2 申请Gitee/GitLab/GitHub帐号 2](#_Toc55674456)

[3 Git操作过程 2](#_Toc55674457)

[3.1 实验场景(1)：仓库创建与提交 2](#_Toc55674458)

[3.2 实验场景(2)：分支管理 8](#_Toc55674459)

[3.3 实验场景(3)：远程分支管理 14](#_Toc55674460)

[4 小结 16](#_Toc55674461)

# 实验要求

实验目标：

1. 了解配置管理工具Git及相应用环境；

2. 熟练掌握Git的基本指令和分支管理指令；

3. 掌握Git支持软件配置管理的核心机理；

4. 在实践项目中使用Gitee/GitLab/GitHub管理自己的项目源代码。

实验要求：

1. 本次实验由个人单独完成；

2. 按照第三条步骤进行实验；

3. 遵循实验报告模板撰写报告；

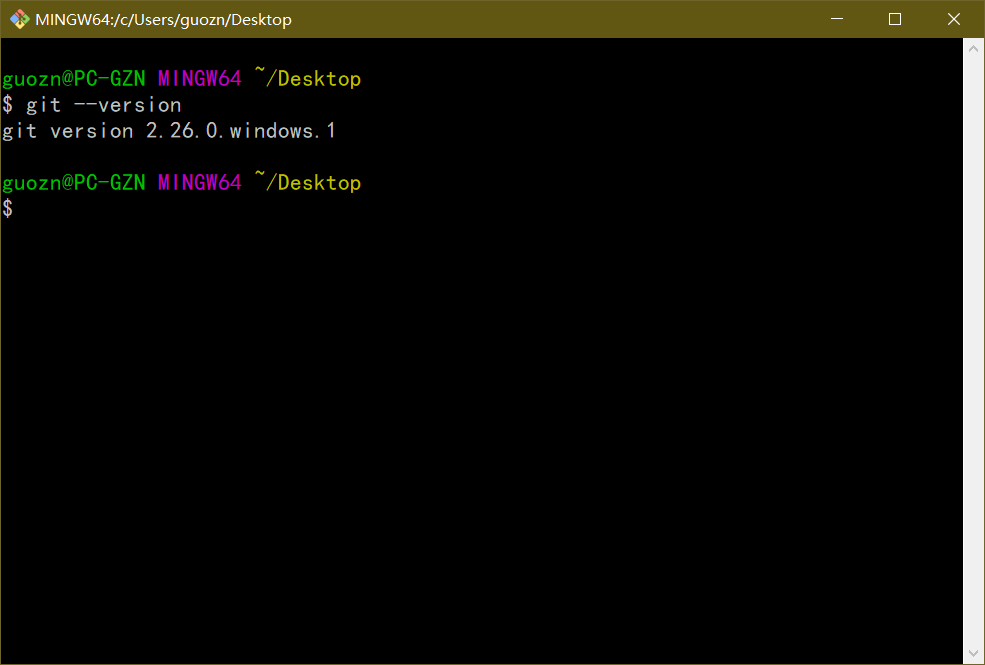
4. 上交文件：

1. 实验报告文件命名：《软件过程与工具》实验2报告-学号-姓名.docx；
2. 上交时间：第9周周六（2020.11.07）24：00前；
3. 上交方式：发送到老师信箱：fgx@hit.edu.cn。

# 安装Git

## 本地机器上安装Git

如果在Windows下安装Git，给出安装的Git版本号和在本地机器上安装Git后的运行界面（给出主要界面即可）。

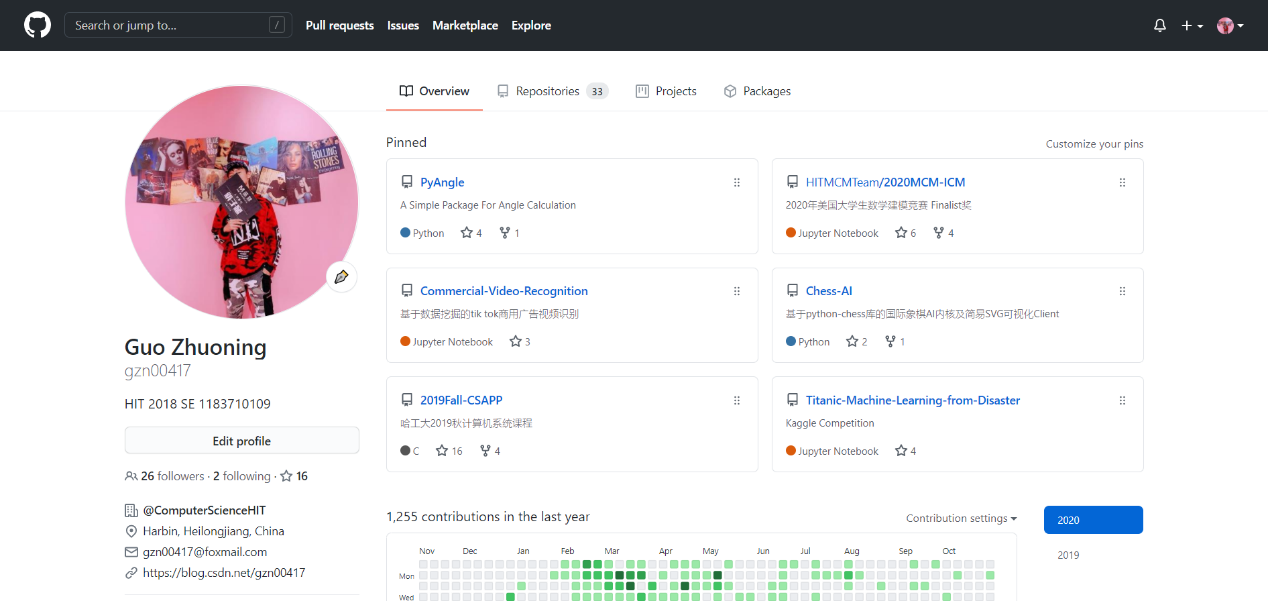


## 申请Gitee/GitLab/GitHub帐号

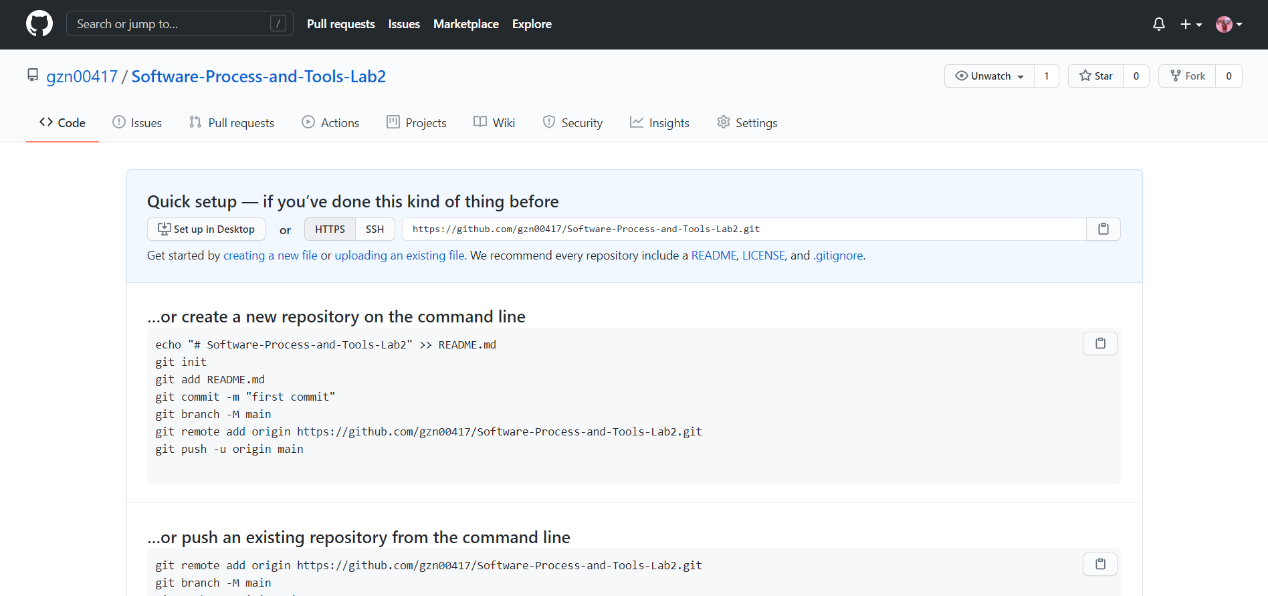
给出自己在Gitee/GitLab/GitHub上申请的帐号名称和本次实验中涉及的项目的URL地址。

给出Gitee/GitLab/GitHub网站上你的账号信息和项目信息的截图。

Gzn00417



https://github.com/gzn00417/Software-Process-and-Tools-Lab2



# Git操作过程

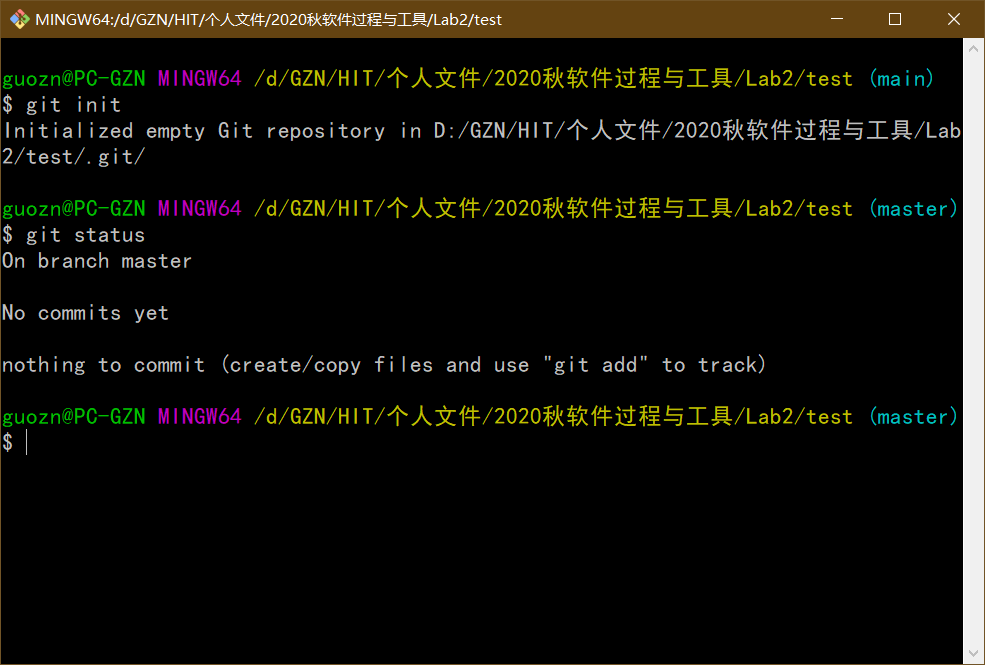
## 实验场景(1)：仓库创建与提交

给出R0~R7的操作命令，并给出执行界面的截图（命令输入界面和结果界面）。

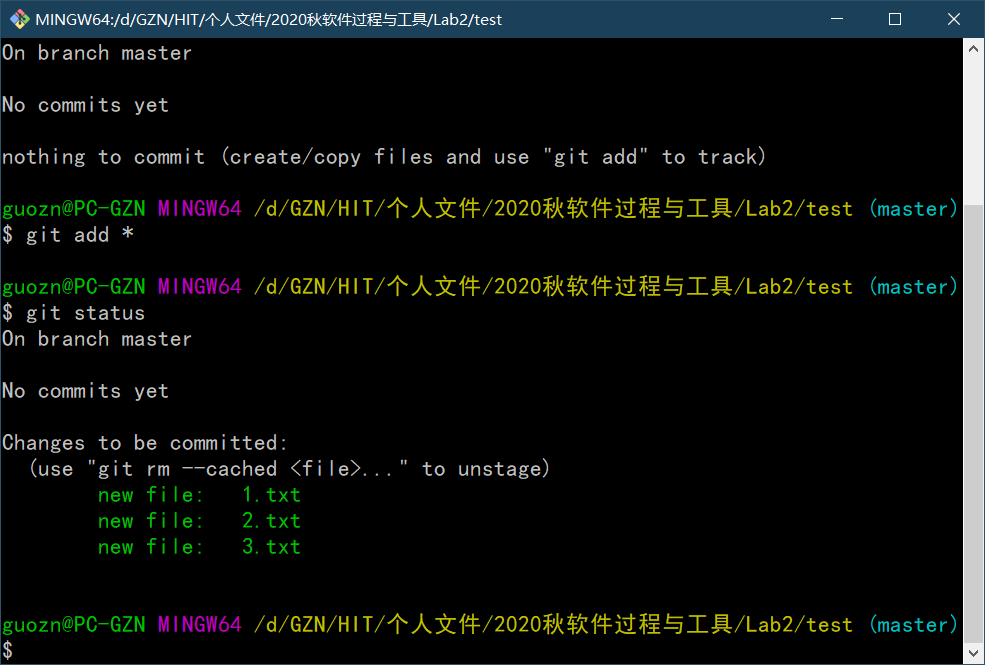
R1:

git init

git status

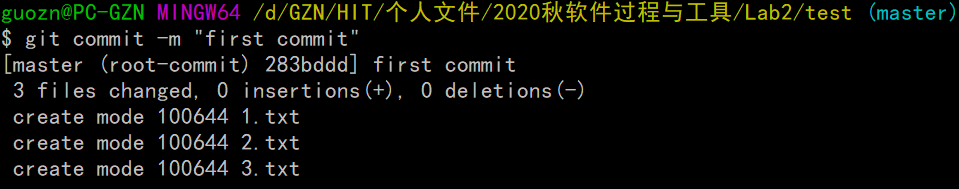


Git add \*



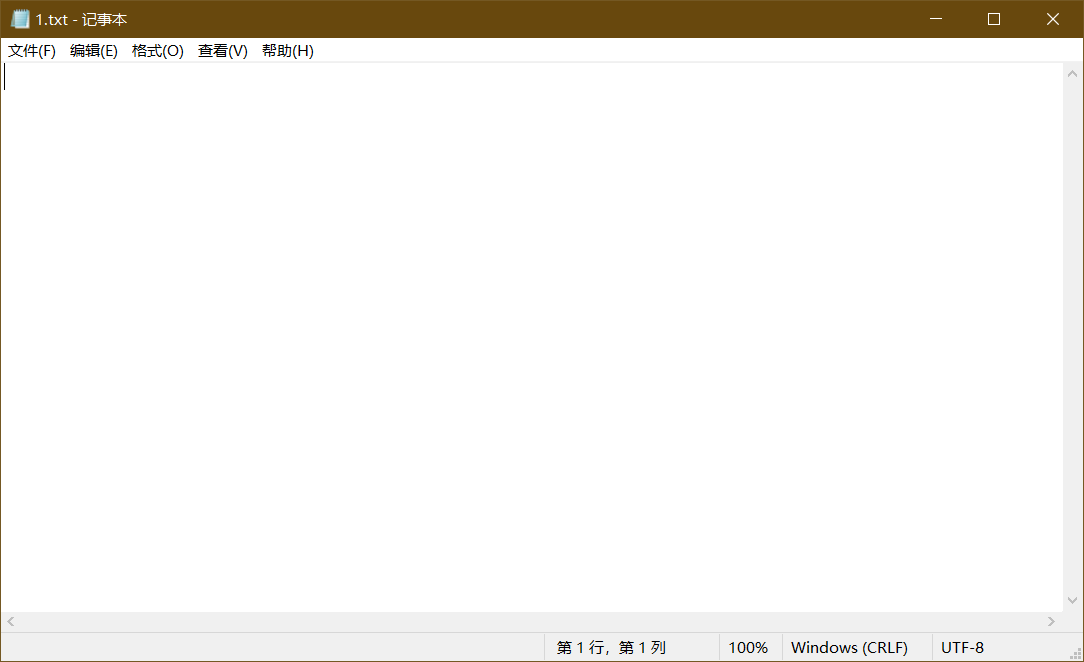
R2：提交文件，查看仓库状态

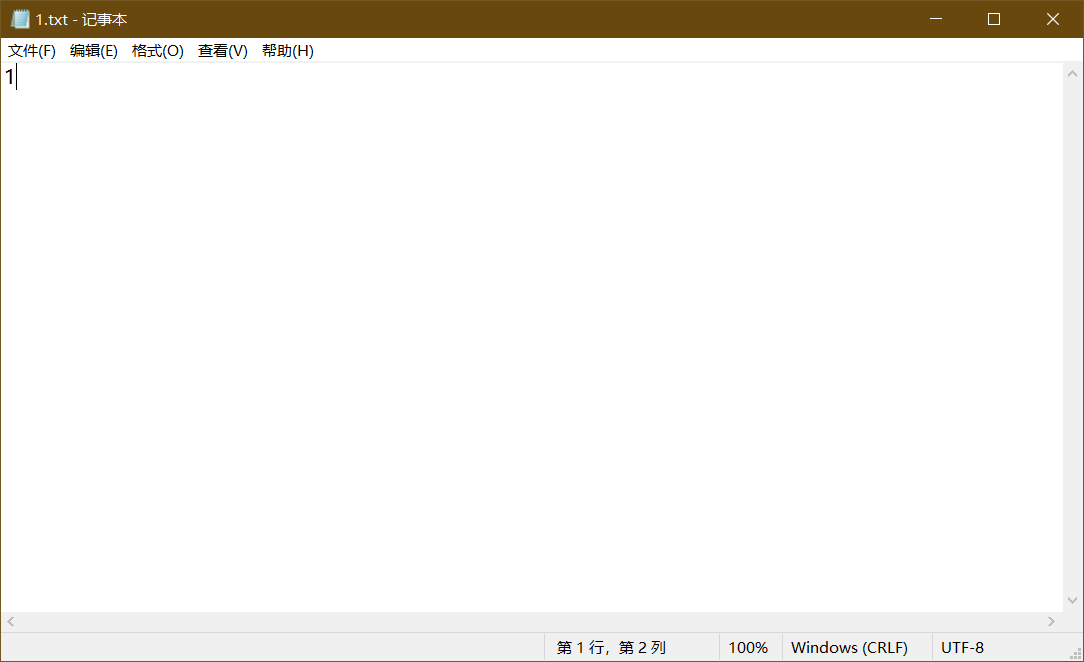
git commit -m “first commit”



R3：

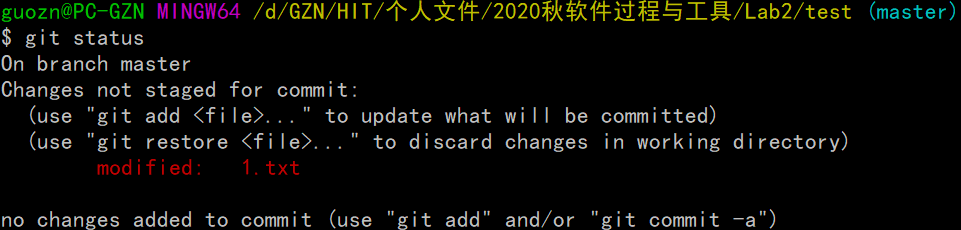
修改1.txt





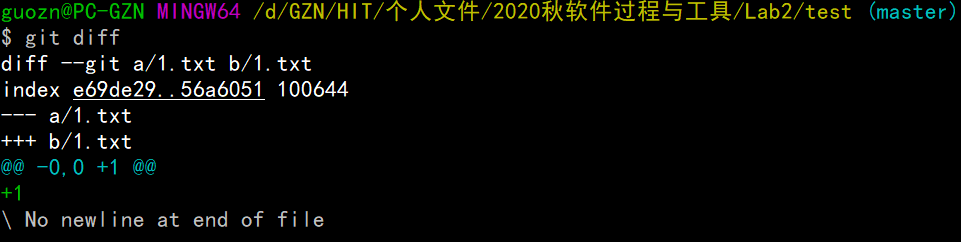
查看仓库状态，有哪些文件被修改

git status



具体修改内容是什么（查看修改后的文件和暂存区域中相应文件的差别）

git diff

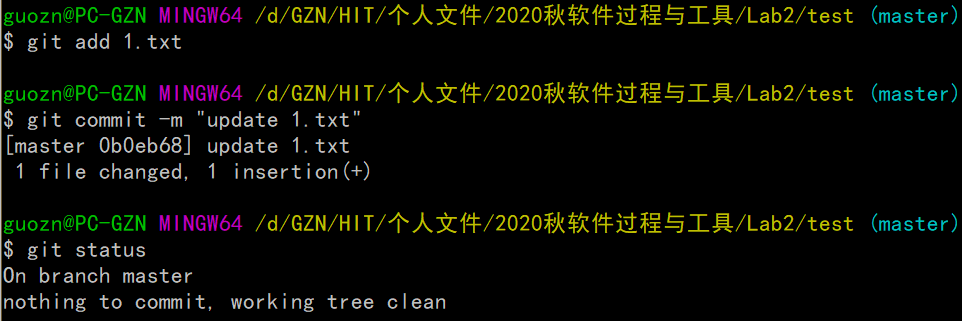


R4：重新提交，并查看仓库状态

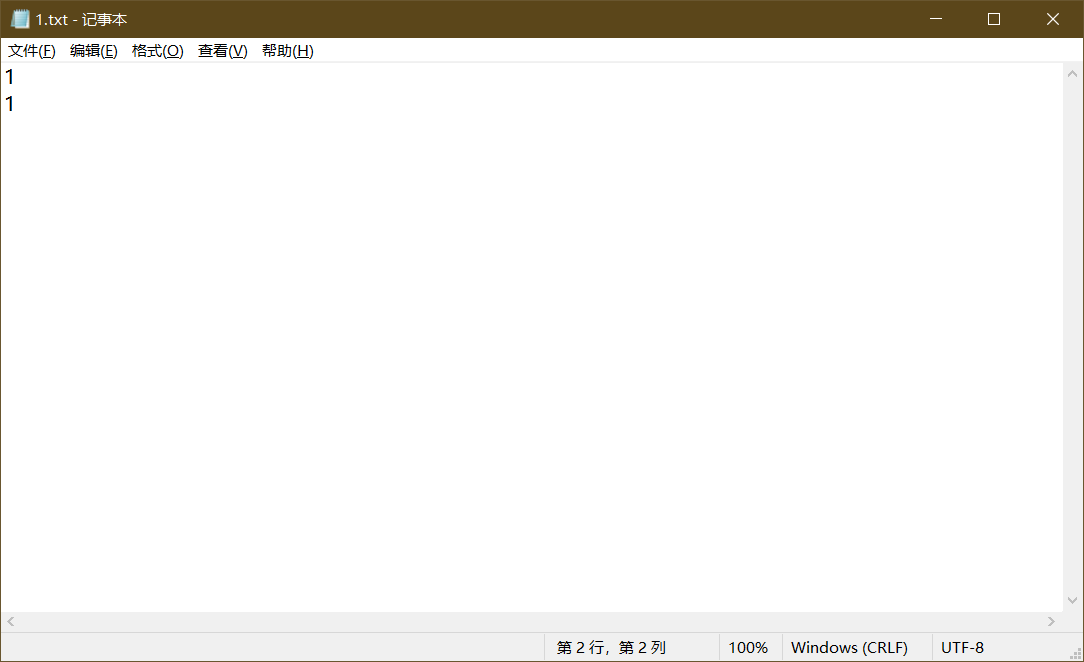
git add 1.txt

git commit -m “update 1.txt”

git status

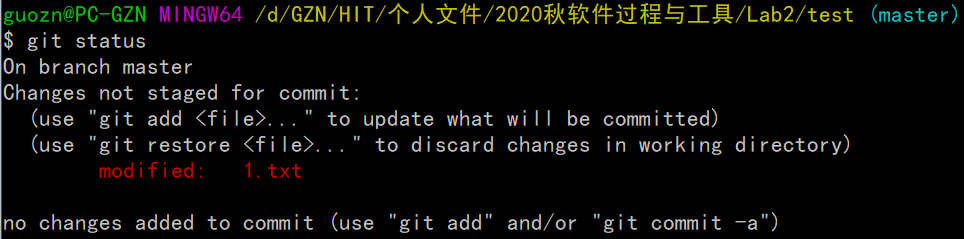


R5：再次修改1.txt



查看仓库状态

git status

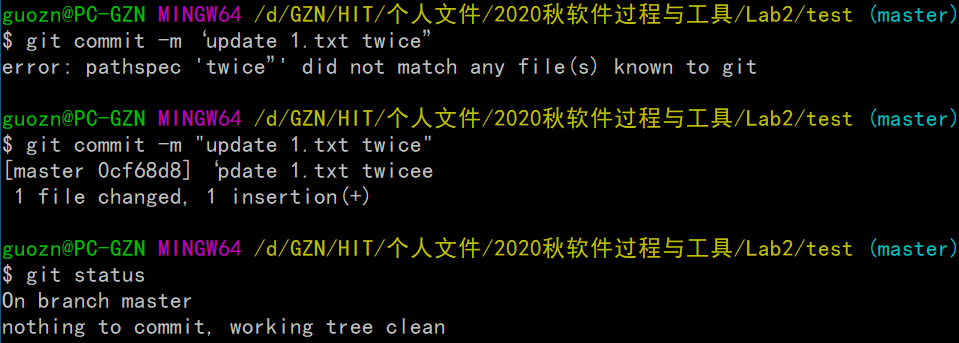


提交

git add 1.txt

git commit -m “update 1.txt twice”

git status



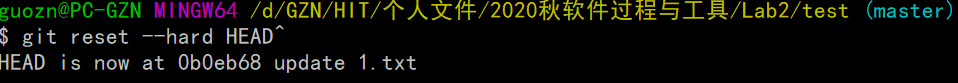
R6：查看提交日志

git log



撤销上次提交

git reset --hard HEAD^

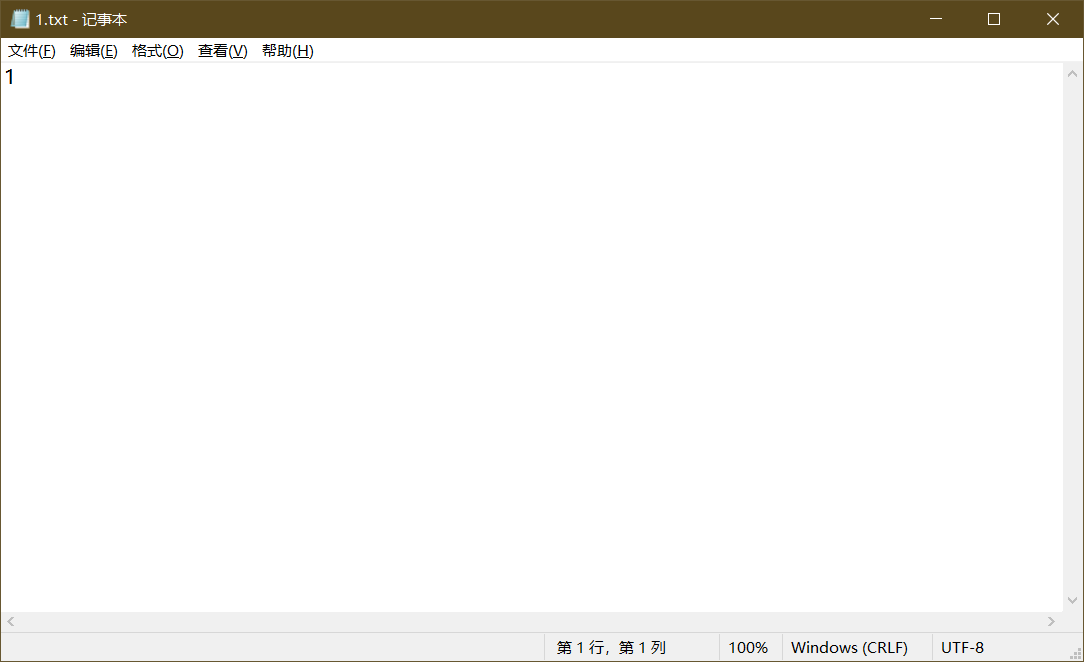


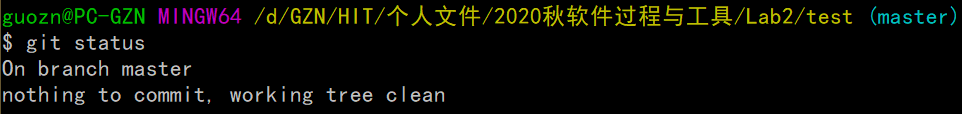
再次查看提交日志

git log



发现最后一次提交没有了，查看1.txt，发现修改也被撤销了





R7：查询提交记录

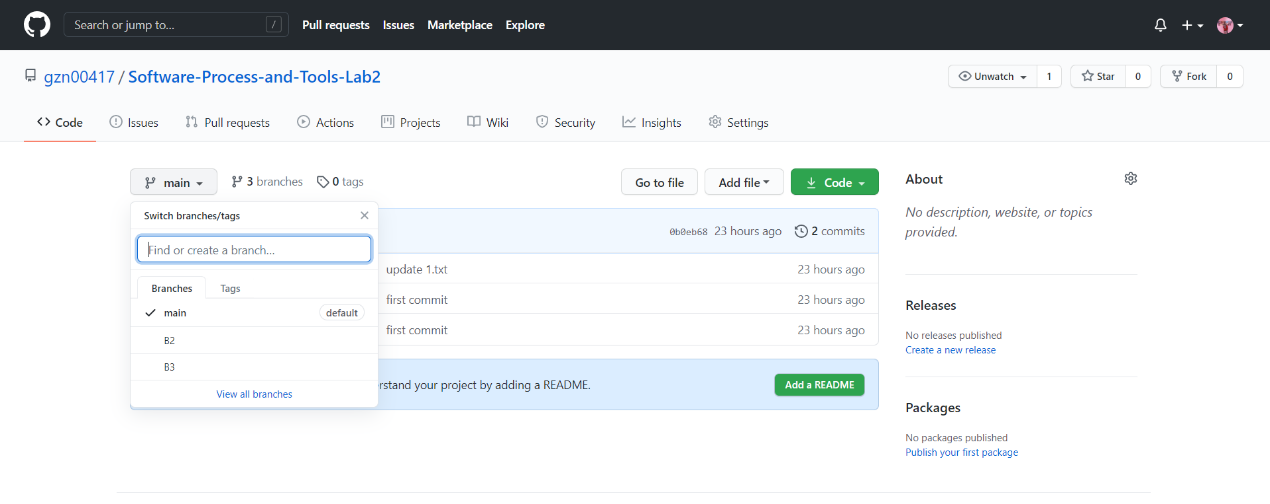
git log



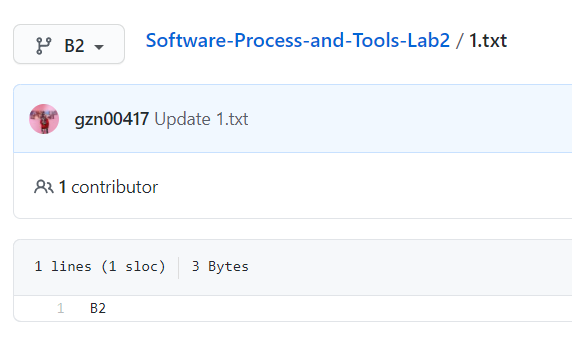
## 实验场景(2)：分支管理

给出R8~R15的操作命令，并给出执行界面的截图（命令输入界面和结果界面）。

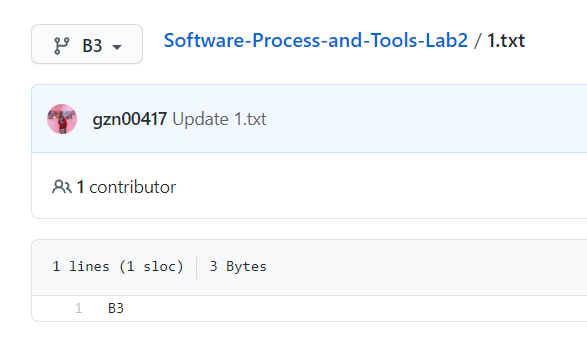
在GitHub仓库中上传所有的资源文件，形成main分支，在此分支上创建B2和B3两个分支



修改B2分支的README.txt如下

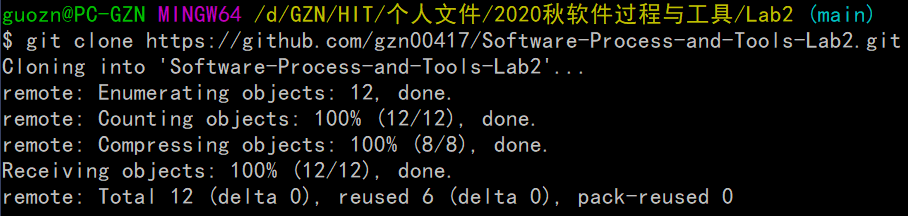


修改B3分支的README.txt如下



R8：从GitHub仓库克隆到本地

git clone https://github.com/gzn00417/Software-Process-and-Tools-Lab2.git

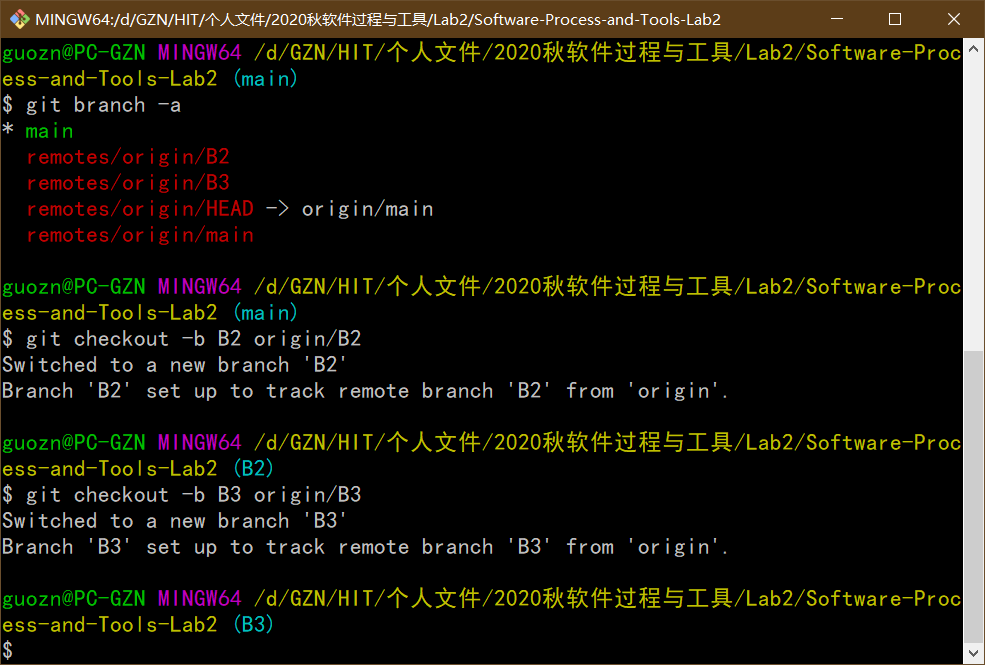


R9：获得该仓库的全部分支

git branch -a

git checkout -b B2 origin/B2

git checkout -b B3 origin/B3



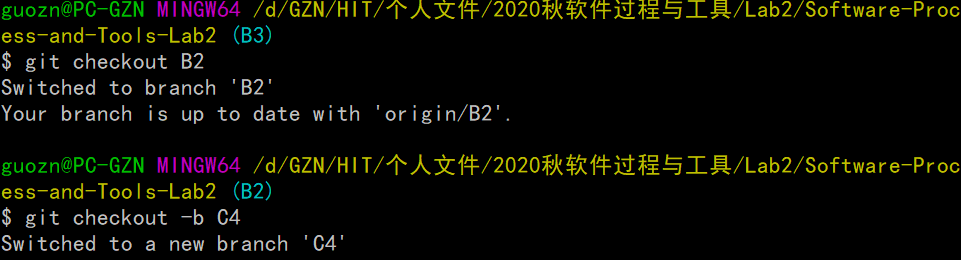
R10：在B2分支基础上创建一个新分支C4

切换工作分支到B2

git checkout B2

新建分支C4

git checkout -b C4

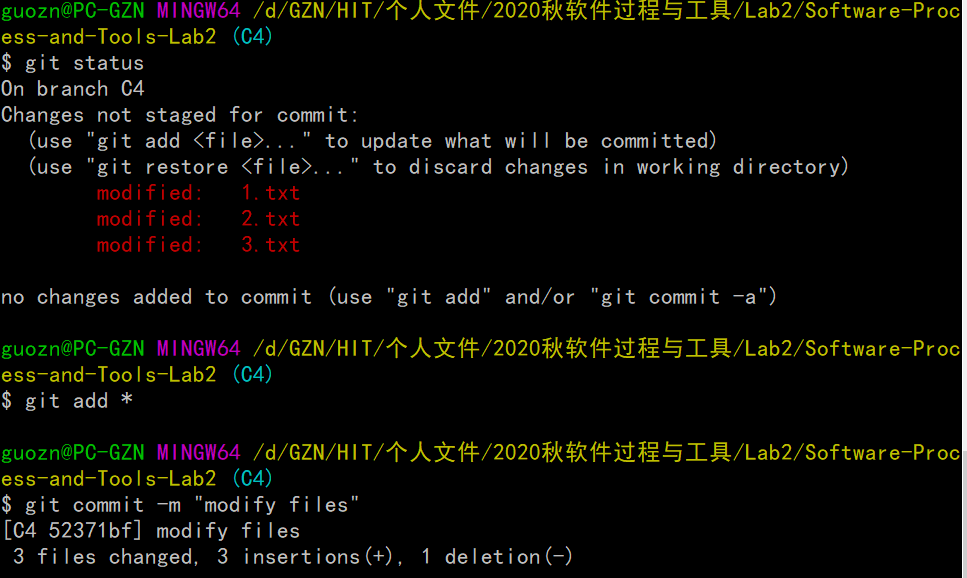


R11：在C4上，对4个文件进行修改并提交

git status

git add \*

git commit -m “modified 4 files”



R12：在B3分支上对同样的4个文件做不同修改并提交

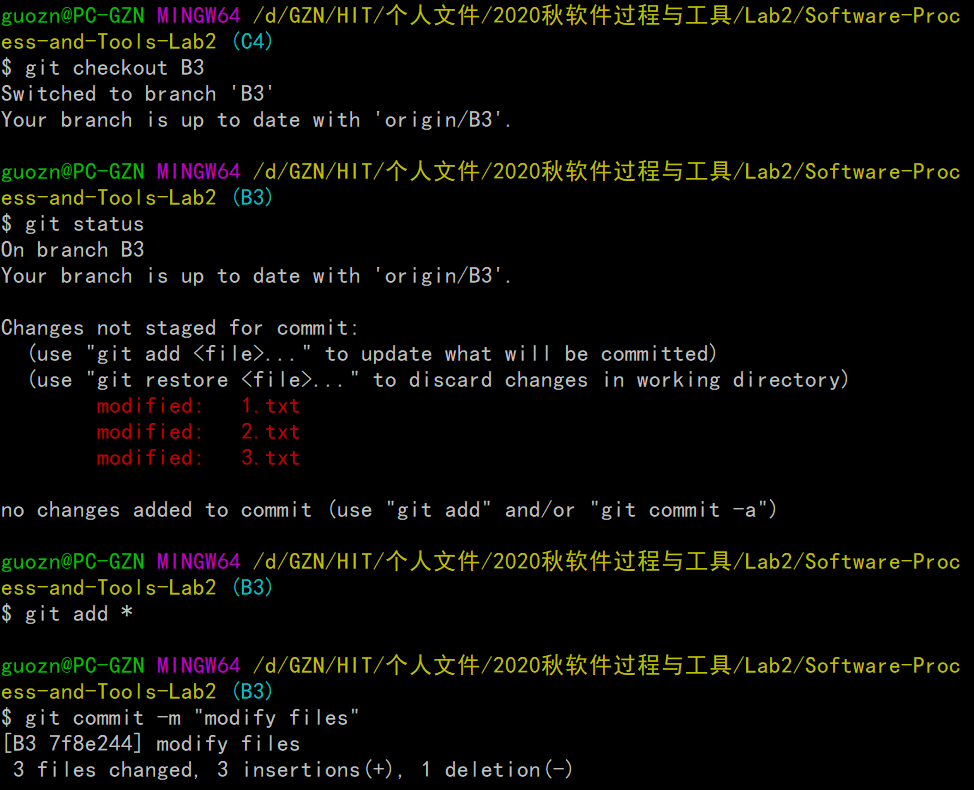
切换工作分支到B3

修改与C4相同的文件，查看仓库状态，提交

git status

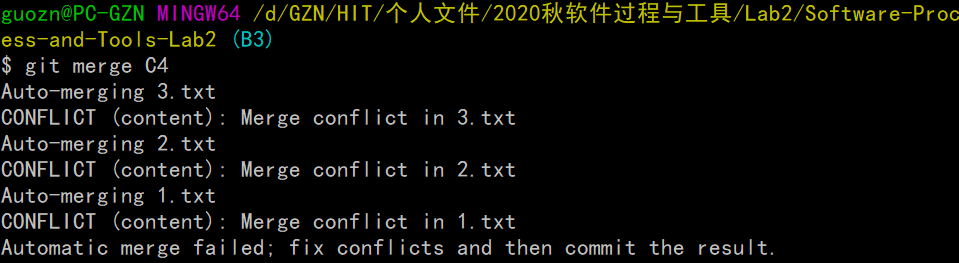
git add \*

git commit -m “ modified 4 files same in C4”



R13：将C4和B3分支合并，若有冲突，手工消解

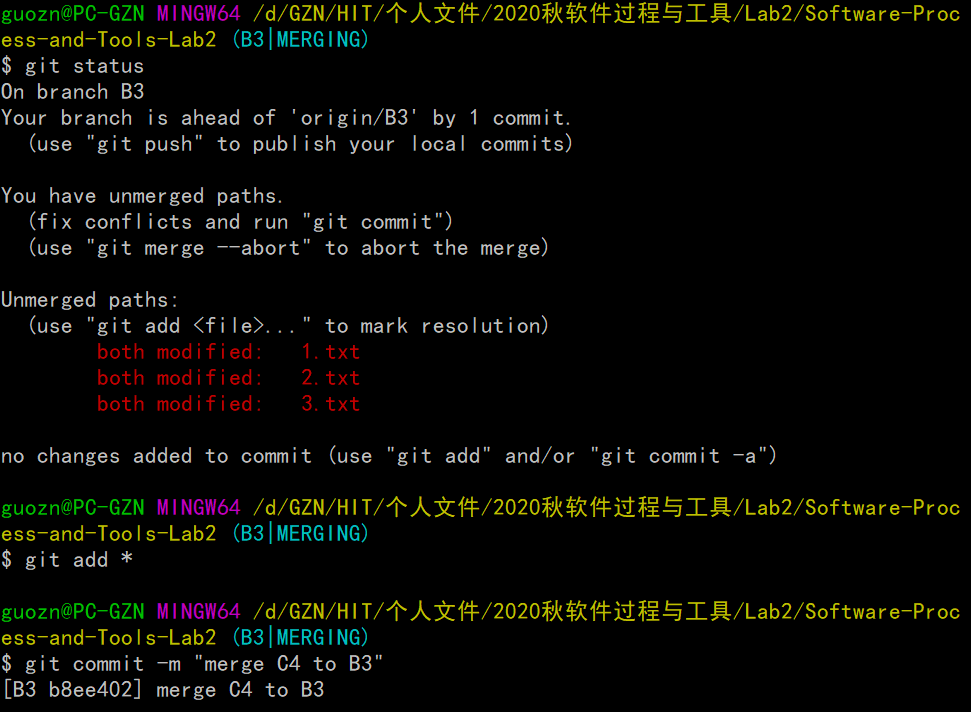
git merge C4



合并分支C4到B3，发生冲突，手工处理后重新提交

git add \*

git commit -m “merge C4 to B3”



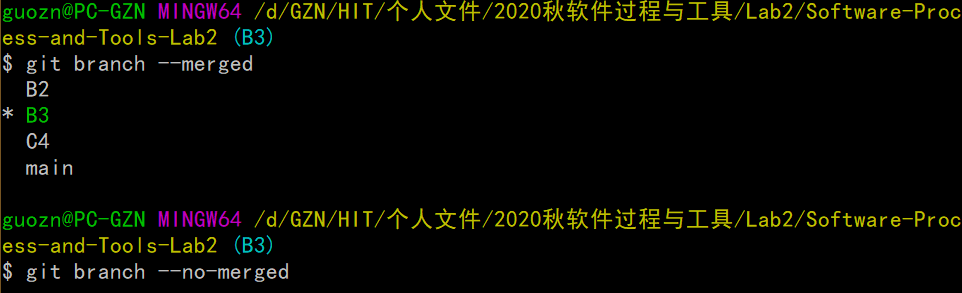
R14：查看目前哪些分支已经合并、哪些分支尚未合并

查看已经被合并的分支

git branch --merged

查看未被合并的分支

git branch --no-merged



R15：将C4和B3合并后的分支删除，将尚未合并的分支合并到一个新分支上，分支名字为你的学号

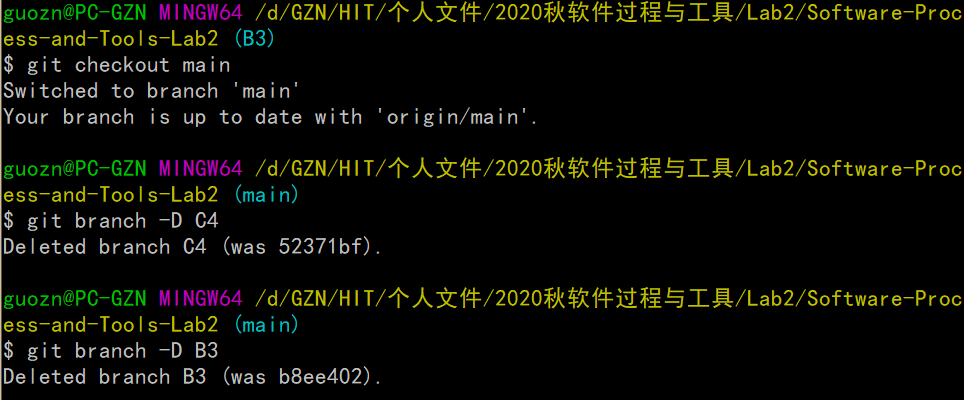
切换到主分支，B1分支，也就是main分支

git checkout main

删除分支B3和C4

git branch -D C4

git branch -D B3

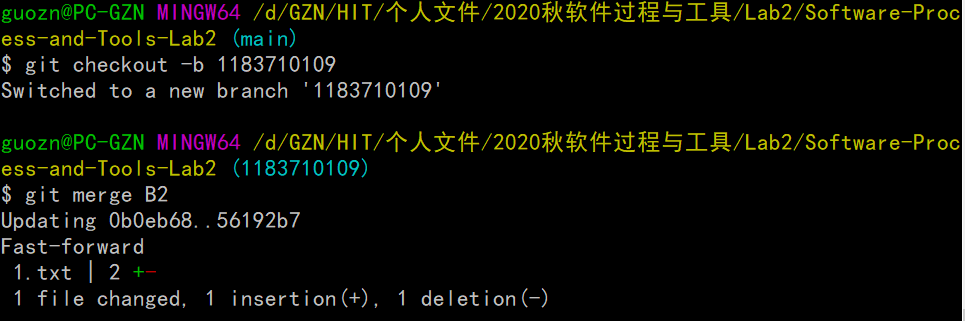


在主分支新建分支1183710109

git checkout -b 1183710109

合并剩余的B2

git merge B2

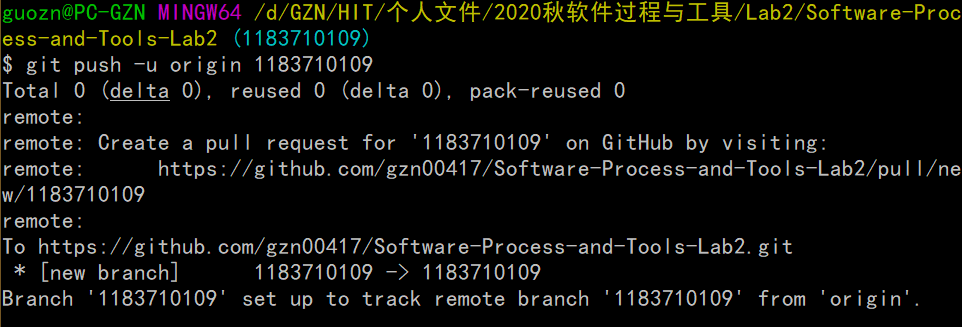


## 实验场景(3)：远程分支管理

给出R16~R18的操作命令，并给出执行界面的截图（命令输入界面和结果界面）。

R16：将本地以你的学号命名的分支推送到Gitee/GitLab/GitHub上

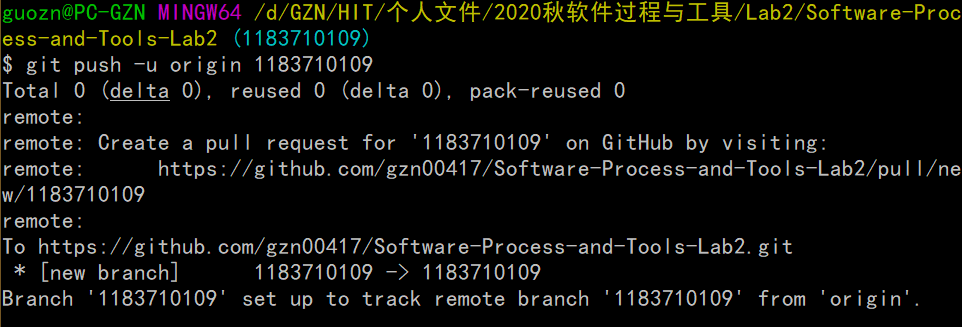
git push -u origin 1183710113



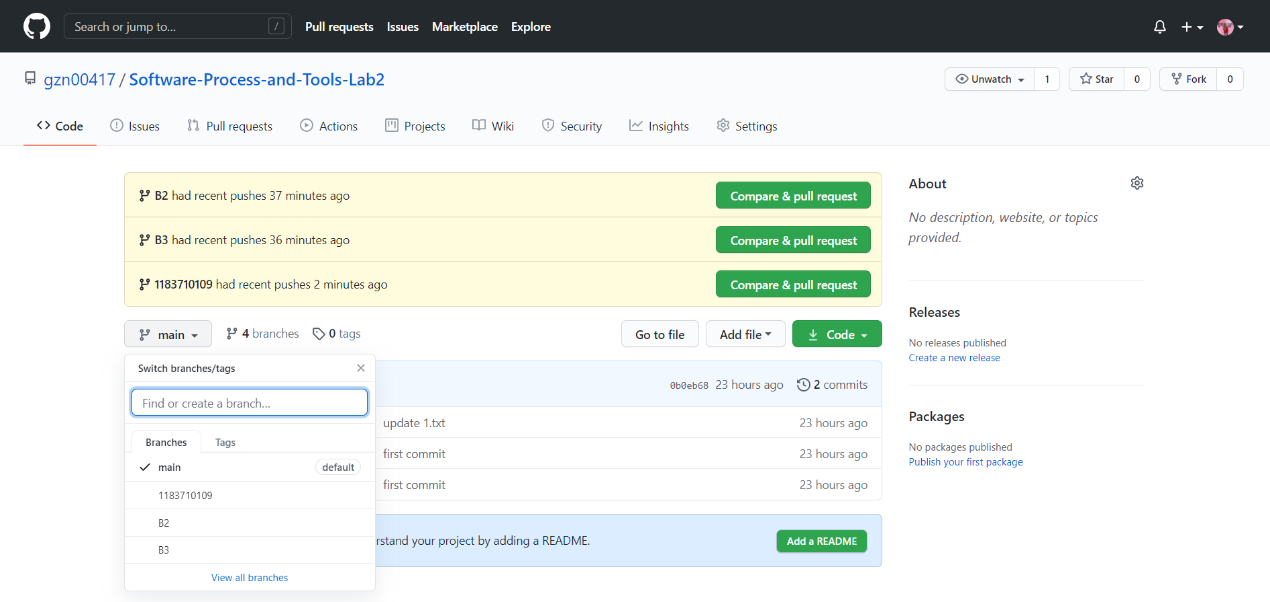
R17：将R1到R7各步骤得到的结果推送到Gitee/GitLab/GitHub上

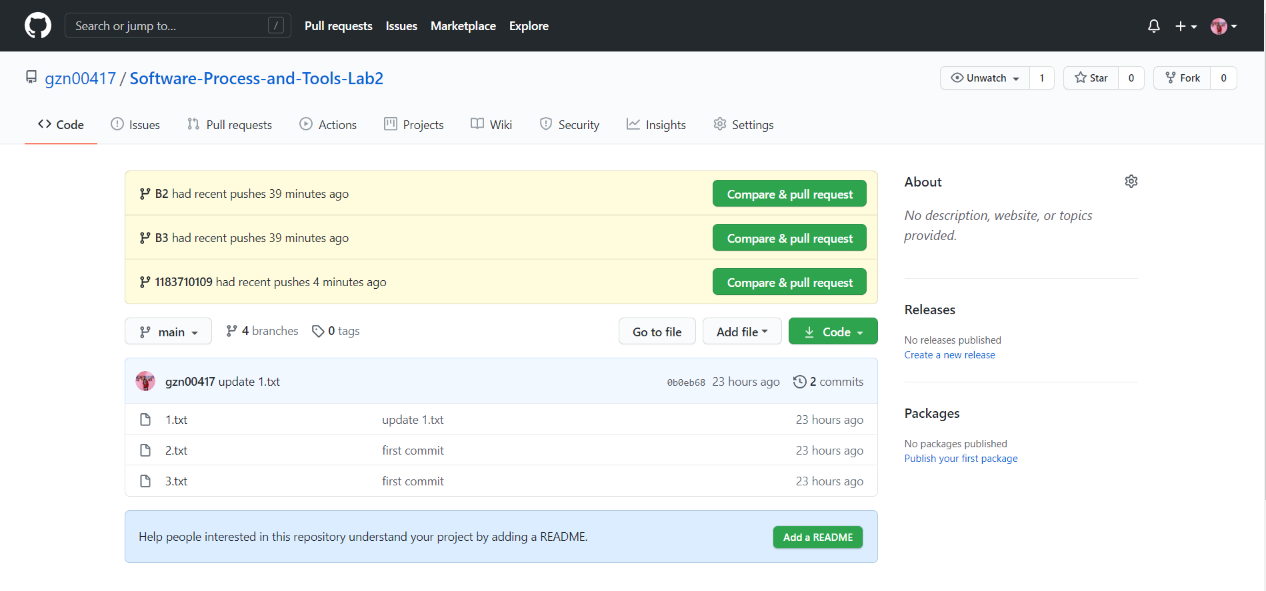
git remote add origin https://github.com/gzn00417/Software-Process-and-Tools-Lab2.git

git push -u origin master



R18：在Gitee/GitHub网站以Web页面的方式查看你的两个仓库的当前状态





# 小结

对本次实验过程和结果的思考：

1. 比较之前的开发经验，使用Git的优点？
2. 在个人开发和团队开发中，Git起到的作用有何主要差异？
3. 之前是否用过其他的版本控制软件？如果有，同Git相比有哪些优缺点？如果没有查阅资料对比一下不同版本控制系统的差别。
4. 在什么情况下适合使用Git、什么情况下没必要使用Git？
5. 其他相关思考。
6. Git的优势在于自动管理代码等文件，避免了开发过程中出现的各种问题，但由于未备份导致版本回退出现困难。极大的方便了开发过程。
7. 个人开发中主要是起到对自己开发过程中产生的文件进行管理，而团队开发中Git主要起到了不同成员间开发的同步问题。
8. 以前并没有用过其他版本控制软件，查阅资料了解了SVN与Git的优缺点。

SVN

优点：管理方便，逻辑明确，符合一般人思维习惯。易于管理，集中式服务器更能保证安全性。代码一致性非常高。适合开发人数不多的项目开发。

缺点：服务器压力太大，数据库容量暴增。如果不能连接到服务器上，基本上不可以工作，如果服务器不能连接上，就不能提交，还原，对比等等。不适合开源开发。

Git

优点：适合分布式开发，强调个体，速度快、灵活。任意两个开发者之间可以很容易的解决冲突。公共服务器压力和数据量都不会太大。

缺点：学习周期相对而言比较长。不符合常规思维。代码保密性差，一旦开发者把整个库克隆下来就可以完全公开所有代码和版本信息。

1. 在代码量较小或者实现功能比较简单的时候没必要使用Git。开发人员较少也没有必要使用Git。因为Git分布式版本管理目的就在于协调团队中各种代码的冲突。而开发团队很大、代码量很大时Git就很有用了，因为人工去协调矛盾效率很低。