**深 圳 大 学 实 验 报 告**

**课程名称： 软件工程**

**实验项目名称： 实验4 配置管理及团队开发**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 卢亚辉**

**报告人： 郑雨婷 学号： 2021150122 班级： 高性能**

**实验时间：2023年10月24日（周二）~2023年11月5日（周日）**

**实验报告提交时间： 2023年11月5日**

**教务部制**

**一、实验要求：**

（1）学习配置管理的基本原理

（2）了解主流的配置管理工具

（3）掌握Git版本管理工具的安装，配置

（4）掌握Git本地库的创建和配置

（5）掌握本地仓库中的文件管理

（6）了解分支和标签概念

（7）掌握如何创建与合并分支

（8）掌握如何解决分支

（9）掌握多人协作进行软件开发的方法

（10）掌握标签的创建和相关操作

**二、实验内容：**

（1）下载安装Git

（2）Git本地仓库的创建和配置

（3）在本地创建一个C语言或者C++项目，将所有的项目源程序文件提交到本地库中。

（4）多次修改某一个源代码文件，进行多次提交。

（5）查看文件的历史记录，并且查看不同版本之间的差别。

（6）将文件恢复到前面的某个版本，给出详细的恢复过程。

（7）删除某一个提交文件，并且查看项目状态。

（8）在本地创建一个C语言或者C++项目，将所有的项目源程序文件提交到本地库中。

（9）为某个文件创建分支，在分支中提交和对比文件

（10）将修改的内容合并到主分支

（11）与项目组同学合作，各自将本地代码提到同一个远程版本库

（12）与项目组同学共同编写同一个代码文件，体验冲突解决。

（13）为项目增加本地和远程标签，并查看、删除标签

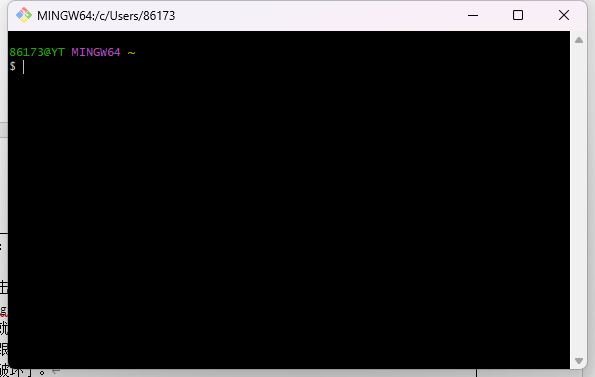
**三、实验步骤**

（1）下载安装Git

从Git官网https://git-scm.com/downloads直接下载安装程序，然后按默认选项安装即可。



安装完成后，在开始菜单里找到“Git”->“Git Bash”，弹出一个类似命令行窗口的东西，就说明Git安装成功！

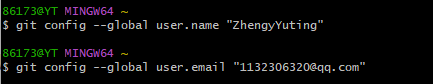


（2）Git本地仓库的创建和配置

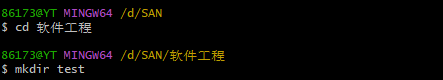
安装完成后，首先我们需要设置用户名和email地址，在命令行输入：

$ git config --global user.name "ZhengYuting"

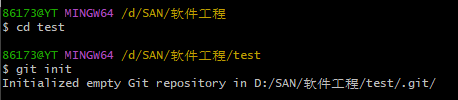
$ git config --global user.email “1132306320@qq.com”



首先，选择一个合适的地方，创建一个空目录：在命令行窗口中使用mkdir指令创建文件夹test。

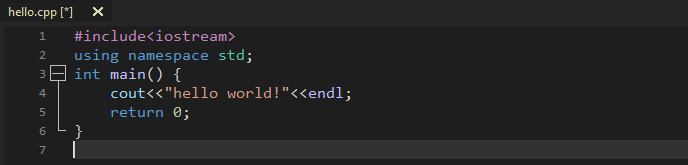


进入到文件夹中，使用git init命令把这个目录变成Git可以管理的仓库，也就是本地仓库。这是一个空的仓库，里面有.git文件，不过已经被隐藏了，这个目录是Git来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。

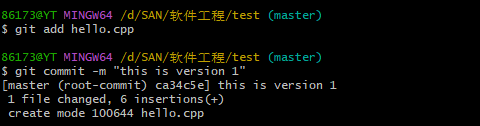


（3）在本地创建一个C语言或者C++项目，将所有的项目源程序文件提交到本地库中。

使用Dev-C++在该文件夹内新建一个cpp文件，本次实验中编写一个简单的hello world程序为例。

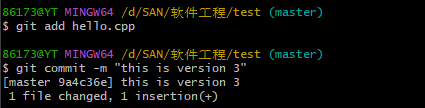
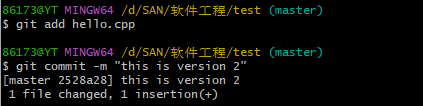


然后在git bash here的命令行窗口使用git add命名把文件添加到git本地仓库。没有错误信息就说明添加成功了，如果有错误信息的话，就需要一步一步慢慢修改但是此时hello.cpp文件只是被提交到暂存区，表示即将被提交到本地仓库。然后使用git commit指令，将暂存区中的文件添加到本地仓库，并且对文件做一个说明并提交说明信息。



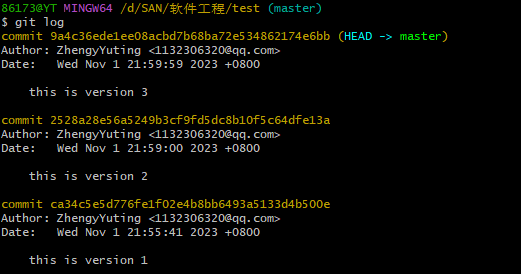
（4）多次修改某一个源代码文件，进行多次提交。

将代码进行修改，规定第几次修改就输出几个hello world,修改过后再次提交。这样修改两次，提交两次，最终总共上传了三个版本。

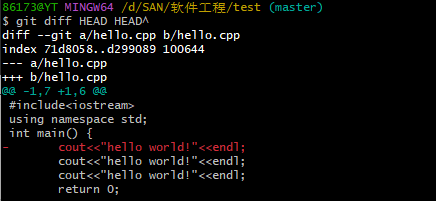


（5）查看文件的历史记录，并且查看不同版本之间的差别。

Git和其他版本控制系统相同，可以查看历史记录。我们用git log命令查看历史记录。git log命令显示从最近到最远的提交日志，我们可以看到3次提交，根据我们的提交说明信息，可以知道最近的一次是version 3，上一次是version 2，最早的一次是version 1。同时也发现，我的提交说明信息写的并不标准，我无法知道到底对文件进行了什么改动，提交说明信息应当将增加了什么或删除了什么言简意赅得写出。



同时，我们可以使用git diff来查看不同版本之间的差别。HEAD是指当前文件，HEAD^是指上一个提交的版本文件。

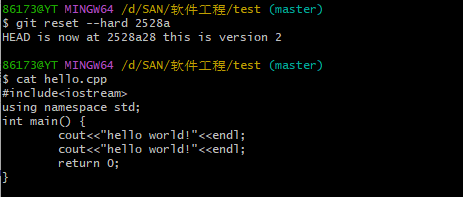


（6）将文件恢复到前面的某个版本，给出详细的恢复过程。

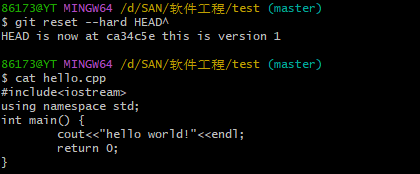
将文件恢复到前面的某个版本,有两种方法。第erzhong种方法是使用git reset --hard + id可以回退到指定id的版本。每个不同的提交的版本都有一个对应的id，这个id也就是git log 中可以查看到的一串数字。



版本号没必要写全，前几位就可以了，Git会自动去找。当然也不能只写前一两位，因为Git可能会找到多个版本号，就无法确定是哪一个了。选第二个版本的前五位2528a，执行git reset --hard 2528a。然后再次查看hello.cpp的内容，确实是输出了两次helloworld.

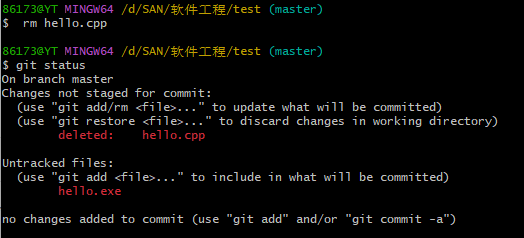


第二种方法用HEAD表示当前版本，上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。执行git reset --hard HEAD^之后查看当前hello.cpp的内容，只输出一次helloworld。这是因为上一步已经将当前文件改为了第二个版本。它的上一个版本是最初的只输出一次的版本。

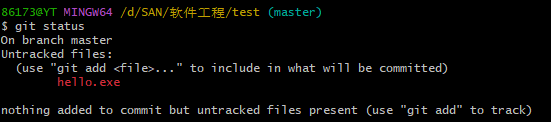
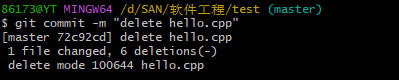


（7）删除某一个提交文件，并且查看项目状态。

使用rm hello.cpp把刚才的文件删除，但是只是删除了工作区的文件，但是本地仓库中的文件还在。使用git status查看项目状态，可以看到hello.cpp文件被删除了。

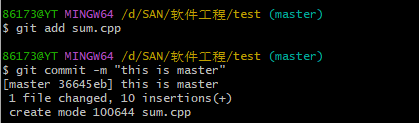
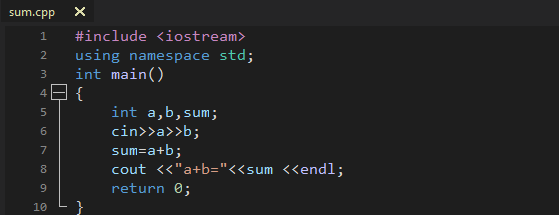


想要从版本库中删除该文件，使用git rm hello.cpp.之后将这次删除提交，这样就真正从删除了。再次用git status查看项目状态。



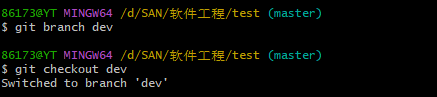
（8）在本地创建一个C语言或者C++项目，将所有的项目源程序文件提交到本地库中。

写一个简单的求和程序，实现输入两个整数a和b，输出它们的和，将文件命名为sum.cpp。然后将其上传到本地库中。

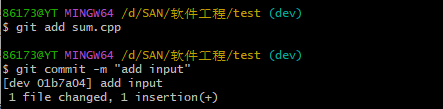
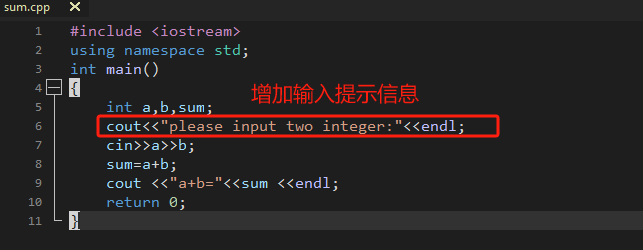


（9）为某个文件创建分支，在分支中提交和对比文件。

Git支持分支管理，，一个分支代表一条独立的开发线。使用分支意味着你可以从开发主线上分离开来，然后在不影响主线的同时继续工作。首先创建分支dev并且切换到该分支，可以使用$ git branch dev和$ git checkout dev先创建再切换，也可以直接git checkout命令加上-b参数表示创建并切换。

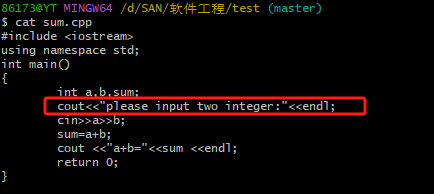
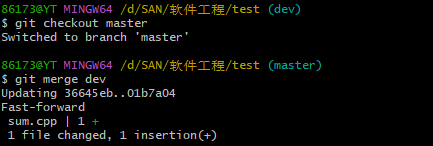


然后，将sum.cpp进行修改,加一行提示输入信息，然后提交sum.cpp，此时是在分支中提交总共被修改过的文件。



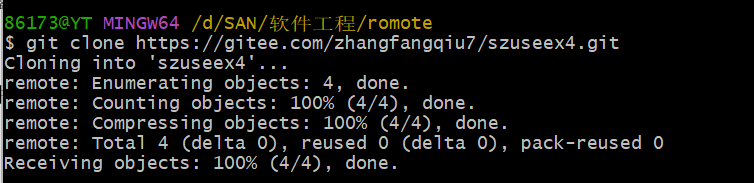
（10）将修改的内容合并到主分支。

首先切换回主分支，再主分支中使用git merge dev，就可以将dev分支的工作结果合并到主分支master上。合并完成后，查看sum.cpp的内容，可以看到是增加了提示信息的，也就是dev分支中的结果，说明合并成功。

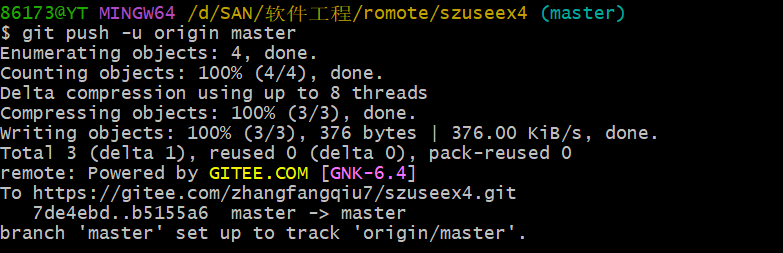


（11）与项目组同学合作，各自将本地代码提到同一个远程版本库。

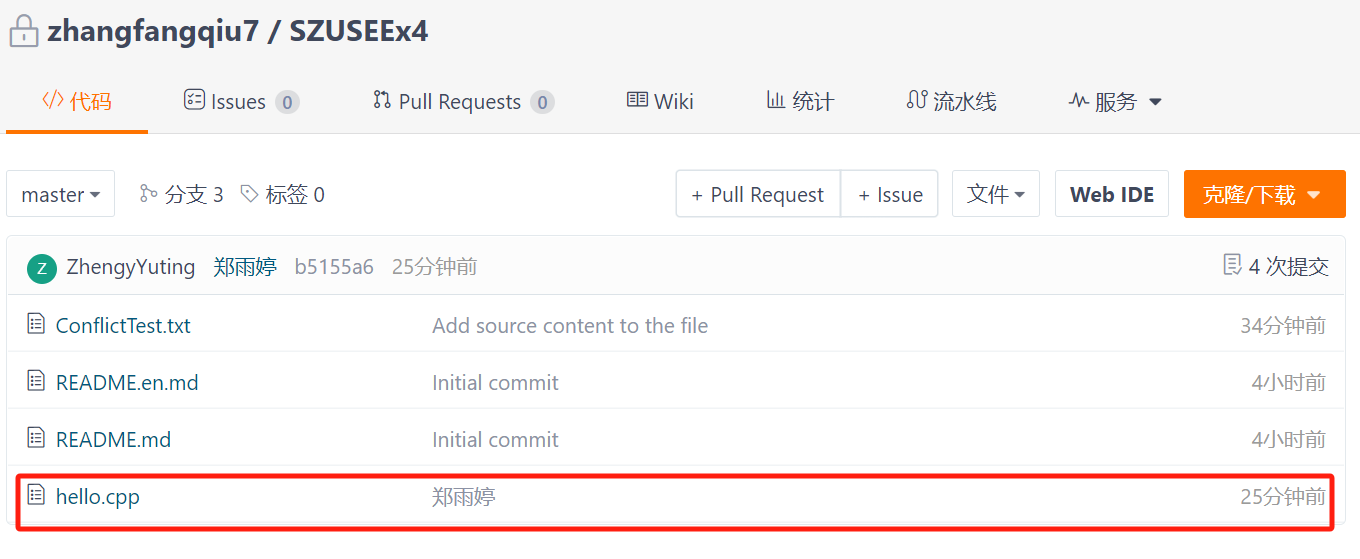
这一步与小组成员共同完成。首先刘雨鑫同学在Gitee创建了一个远程仓库，邀请我们成为仓库成员，用git clone将远程仓库克隆到本地。



然后用git push将本地分支推送到远程仓库。

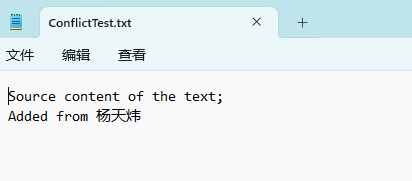
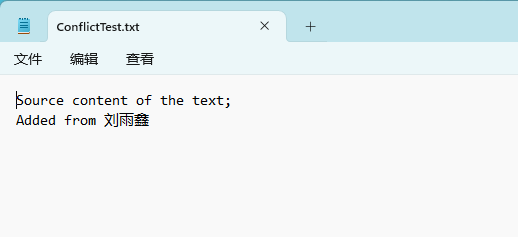


打开远程仓库，可以看到我新增的文件hello.cpp,其它文件是小组的其他成员上传的。

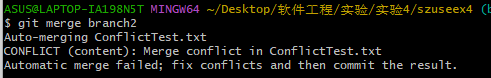


（12）与项目组同学共同编写同一个代码文件，体验冲突解决。

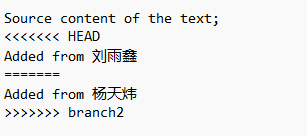
冲突是两个不同分支改同一文件的时候产生的,在master分支添加了一个ConflictTest.txt。一个同学在branch1分支里面加入了"Added from 刘雨鑫" ，另一个同学在branch2里面的ConflictTest.txt里面加一句"Added from 杨天炜"，然后push上去。



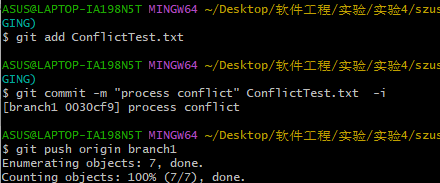
此时，使用merge指令对两个分支进行合并，发现报出了冲突



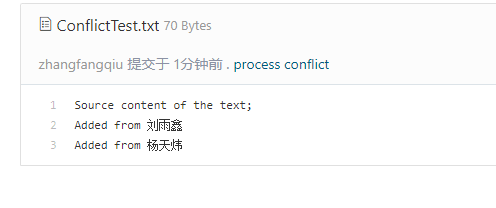
打开文件查看，发现出现了一些意料之外的内容，应当删除它们



最后，通过-icommit方式告知git，已经处理好了冲突问题，并依次交入暂存区，本地库，远程库。

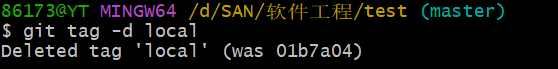
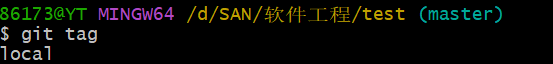


最后，在Gitee的远程库中，我们可以看到我们想要的结果



（13）为项目增加本地和远程标签，并查看、删除标签。

在本地Git中打标签，首先切换到需要打标签的分支上。git tag就可以打一个新标签，默认标签是打在该分支最新提交的commit上的。用Git tag来查看标签，用git tag -d 删除标签。



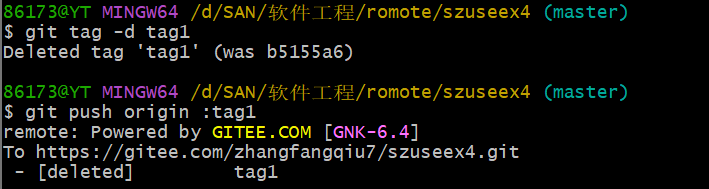
增加远程标签，先在本地上增加标签，然后把tag1 push到远程仓库中



用git ls-remote --tags origin查看远程标签，确实有tag1。



删除远程标签，现在本地删除标签，然后push



**四、实验心得**

本次实验是一次十分新奇的实验，首次使用Git。本次实验，首先对Git版本管理工具的安装和配置。之后我学习了Git本地库的创建和配置。通过实验，我创建了本地仓库、掌握了文件的版本管理和提交。在实验中，我学会了如何将项目源程序文件提交到本地库中，进行多次修改和提交，并查看文件的历史记录。这有助于我更好地跟踪项目的发展和修改。此外，我还了解了分支和标签的概念，以及如何创建和合并分支。分支管理是Git的一个关键特性，通过实验，我学会了如何为某个文件创建分支，在分支中进行提交和对比文件，然后将修改的内容合并到主分支。这对于多人协作项目非常有用。我还体验了多人协作进行软件开发的方法。通过将本地代码提交到远程版本库，并与项目组同学协同工作，我更深入地理解了团队合作和代码协同开发的重要性。最后，我学会了如何为项目增加本地和远程标签，并查看、删除标签。标签是对项目里程碑的重要标记，实验让我掌握了如何有效地使用它们。

总的来说，这次实验不仅帮助我学会了Git版本管理工具的使用，还提高了我的团队合作和配置管理技能。

深圳大学学生实验报告用纸

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2021年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。