

# 操作系统lab0实验报告

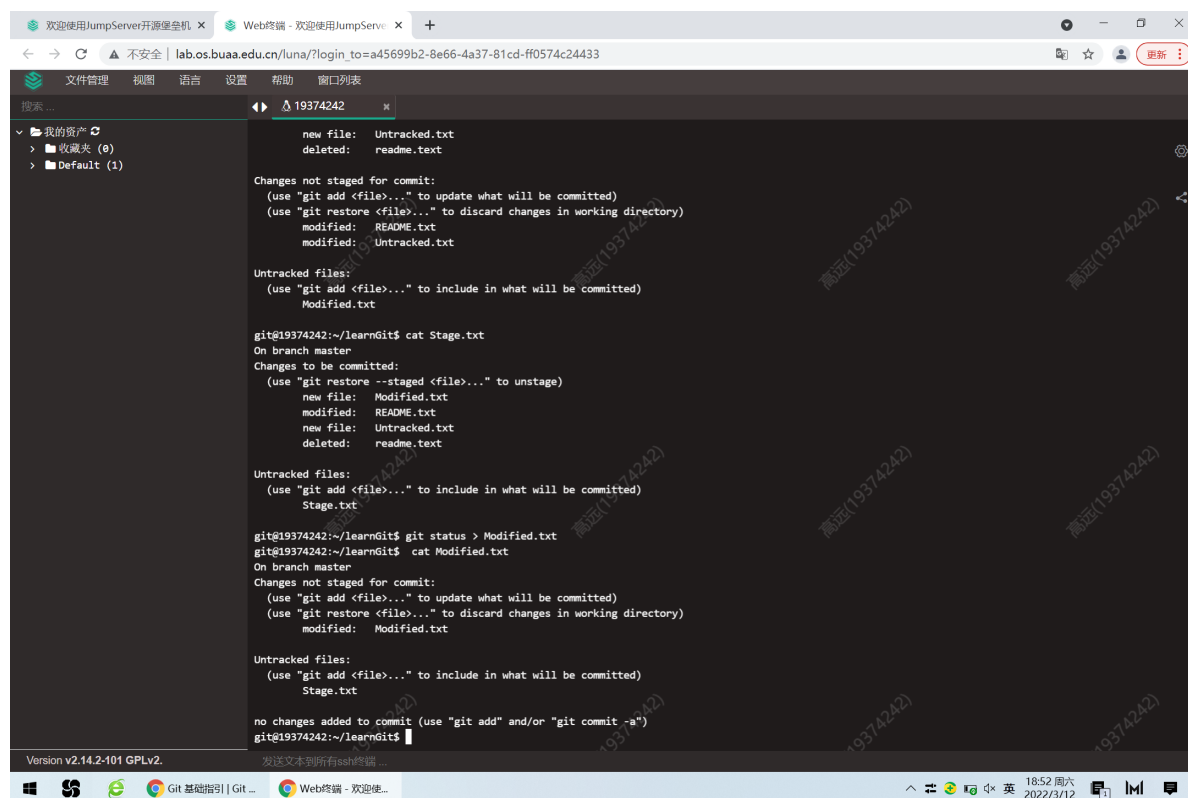
姓名：高远 学号：19374242 班级：202113

## 实验目的

- 1.认识操作系统实验环境
- 2.掌握操作系统实验所需的基本工具

## 实验思考题

### Thinking 0.1:



The screenshot shows a terminal window with the following content:

```
new file:   Untracked.txt
deleted:    readme.txt

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:   README.txt
        modified:   Untracked.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Modified.txt

git@19374242:~/learnGit$ cat Stage.txt
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Modified.txt
        modified:   README.txt
        new file:   Untracked.txt
        deleted:    readme.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Stage.txt

git@19374242:~/learnGit$ git status > Modified.txt
git@19374242:~/learnGit$ cat Modified.txt
On branch master
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:   Modified.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Stage.txt

no changes added to commit (use "git add" and/or "git commit -a")
git@19374242:~/learnGit$
```

如图所示，在使用cat Modified.txt后，它和第一次add之前的status不一样，因为在git add之后，git status的状态会发生变化，add前的存在了Untracked.txt中，之后的存在了stage.txt中，多了一个追踪的文件。同时，在修改过README.txt之后，git的status会发生变化，存在了modified.txt中，因此不一样。

### Thinking 0.2:

add the file 对应的是git add命令，statge the file 对应的是git add命令，commit对应的是git commit命令。

## Thinking 0.3:

- 1.小明不小心删除printf.c文件后可以用git checkout -- printf.c恢复代码文件。
- 2.使用git rm printf.c后可以使用git reset HEAD printf.c, 然后再使用git checkout -- printf.c来恢复, 或者直接使用git checkout HEAD printf.c来恢复。
- 3.想要不从工作区删除也不被提交的方法是git rm --cached printf.c。

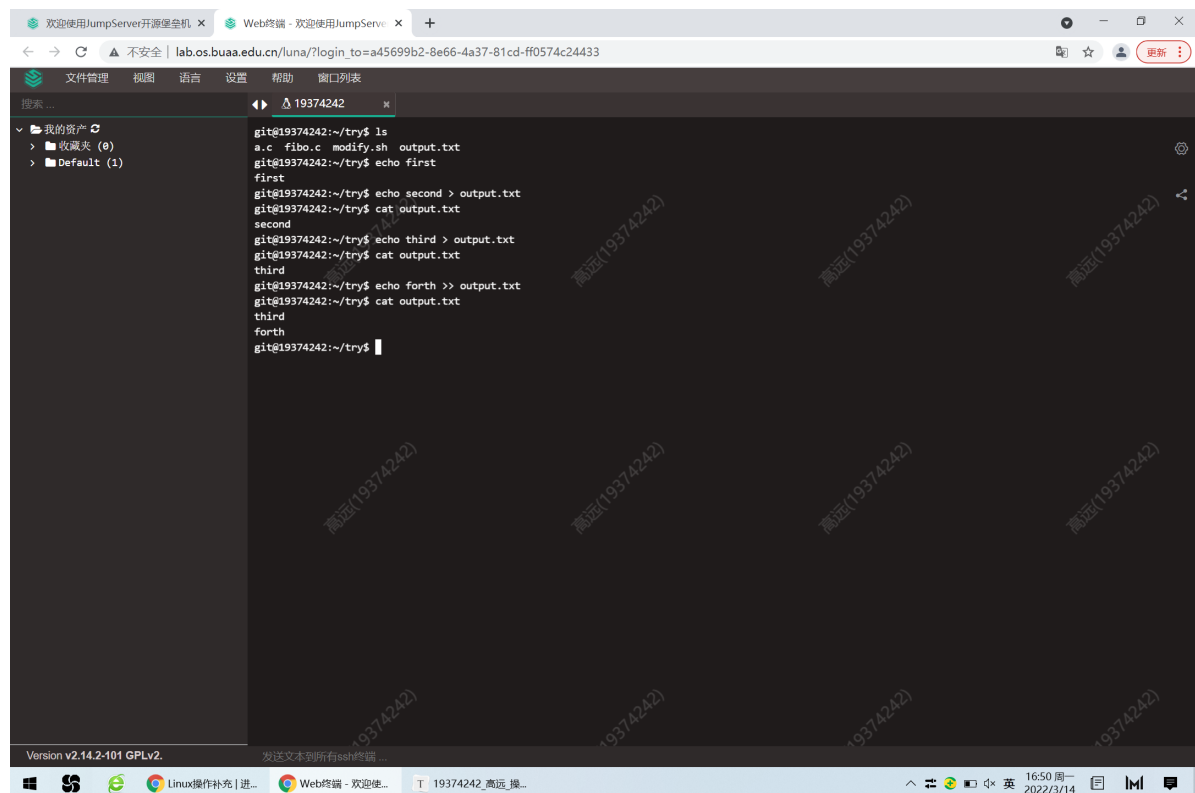
## Thinking 0.4:

在使用git reset --hard HEAD^后第三次提交记录没了, 使用git reset --hard 第一次哈希值后第二次第三次记录都没了, 使用git reset --hard 第三次哈希值后三次记录都恢复了。这条指令让我们可前进, 可后退, HEAD^指回退上一次版本, 也可替换为某一次版本的哈希值, 来实现跳转至某一版本。

## Thinking 0.5:

1错误, 并不是所有分支均被克隆, 只克隆远程库的master分支。2正确, 只有在git push之后才会将本地版本库文件和远程版本库更新。3正确, 只有远程版本库HEAD指向的分支被克隆。4正确, 克隆后工作区的默认分支处于master分支。

## Thinking 0.6:



```
git@19374242:~/try$ ls
a.c  fibo.c  modify.sh  output.txt
git@19374242:~/try$ echo first
first
git@19374242:~/try$ echo second > output.txt
git@19374242:~/try$ cat output.txt
second
git@19374242:~/try$ echo third > output.txt
git@19374242:~/try$ cat output.txt
third
git@19374242:~/try$ echo forth >> output.txt
git@19374242:~/try$ cat output.txt
third
forth
git@19374242:~/try$
```

如图, 输出分别是first, second, third, third forth。

## Thinkint 0.7:

command文件内容



# 数学作业纸

班级:

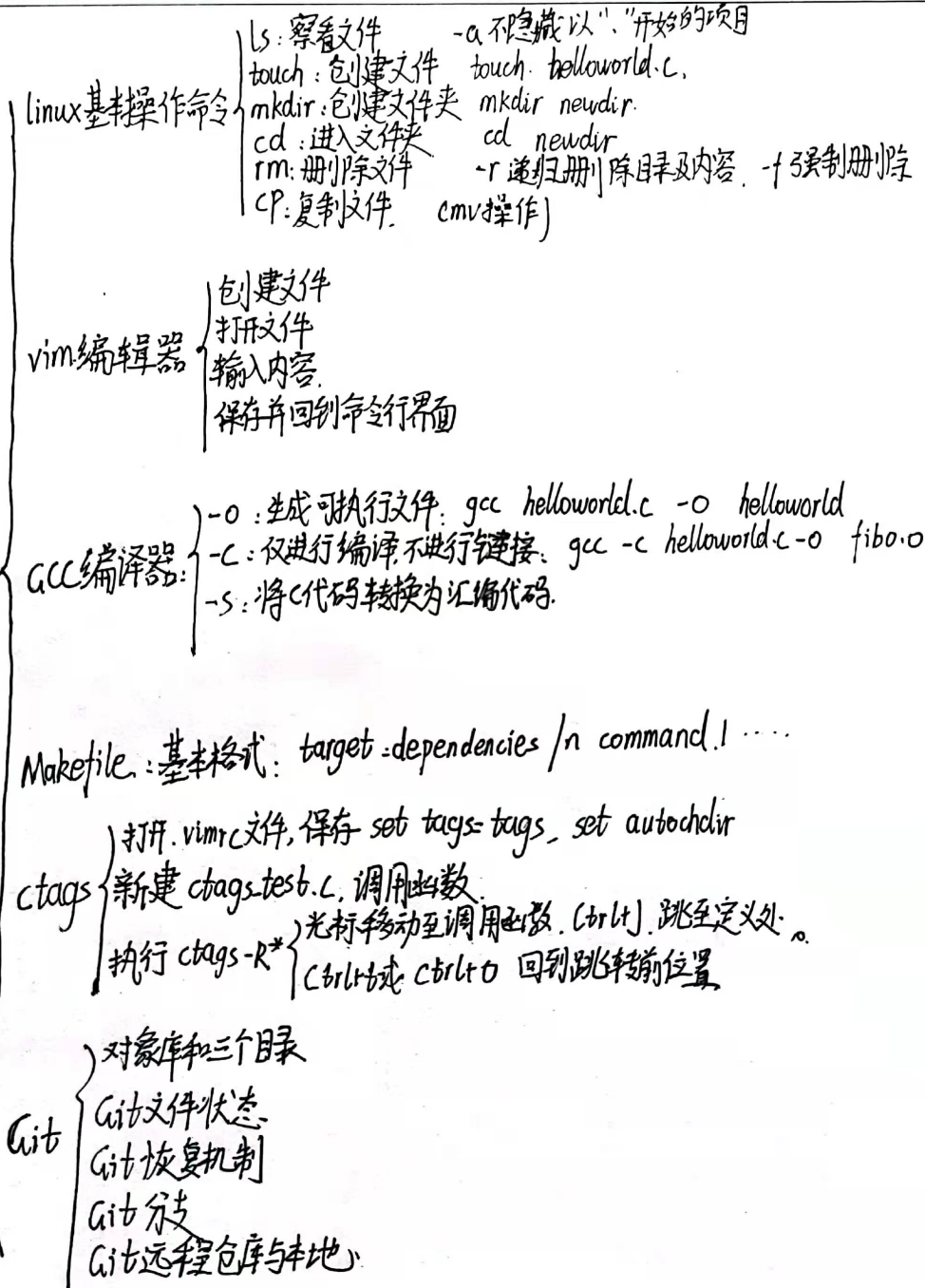
姓名:

编号:

科目:

第 页

实验  
难点



1. 了解linux基本操作命令

2. 学会使用vim来编辑代码

3. 学会使用gcc来进行编译

4. 学会makefile的基本操作, 需要注意的是在执行makefile时, 若使用cd ./code命令, 在命令执行完后会默认回退到当前目录, 所以需要使用&&同时执行命令。

5. 了解ctags的基本功能

6. 初步理解和掌握git的相关功能

7.学会编写shell脚本，理解重定向和管道的概念，需要注意的是在使用管道和重定向命令时，需要注意>指清楚文件，并将标准输出写入文件，而>>是在已有文件的基础上将标准输入写入文件。与此同时，切记akw...\$1>\$1这样的命令是错误的，因为>会先将\$1清空，然后再执行前面的awk，此时awk在对空文件进行操作，产生错误。

8.掌握代码的提交方法

## 体会与感想

---

1.本次lab0实验我花了大约一天时间才全部完成，按理说lab0难度并不大，但是我却做的比较痛苦。经过我的反思，一方面是因为寒假预习不够充分，导致很多操作必须现学，另一方面是因为学习缺乏章法，很多时候都不知道该怎么做，做了为什么会出错，只能依靠百度和询问助教进行纠错，从下个实验开始，我会尽量调整学习方法，做到事半功倍的效果。

2.总的来说，lab0让我学到了很多知识，它不仅让我学会了操作系统实验所需工具的基本使用方法，还让我养成了遇到困难积极查资料或者询问助教求解的方法，使我受益匪浅。

## 残留难点及指导书改进建议

---

我对Git对象库，目录和文件状态的概念还是有一点模糊，指导书对这一块的讲解太重理论性，使我对其实际操作一知半解，希望指导书能够在给出理论解释的同时，进行实际操作演示，这样可以让我们更好地了解理论知识背后的实际意义，从而更好地掌握这块内容。

## 实验结论

---

通过本次实验，我熟悉了操作系统的实验环境，掌握了实验所需工具的基本使用方法，为接下来的学习打下了一个良好的基础，使我受益匪浅。