Lab0 实验报告

——21371163 洪陈天一

Thinking

0.1

README.txt 在三次运行 git status 时所处的位置

- Untracked：“未跟踪的文件”
- Stage：“要提交的变更-新文件”
- Modified：“尚未暂存以备提交的变更-修改”

执行 `git add` 命令后，文件的状态从 未跟踪的文件 变为 要提交的变更中的新文件

提交并再次修改后，文件的状态变为 尚未暂存以备提交的变更中修改过的文件，与第一次的状态并不一致，因为第一次的文件并没有被 git 跟踪，而第三次的文件为跟踪后已修改未暂存。

0.2

- Add the file: `git add <file>`
- Stage the file: `git add <file>`
- Commit: `git commit`

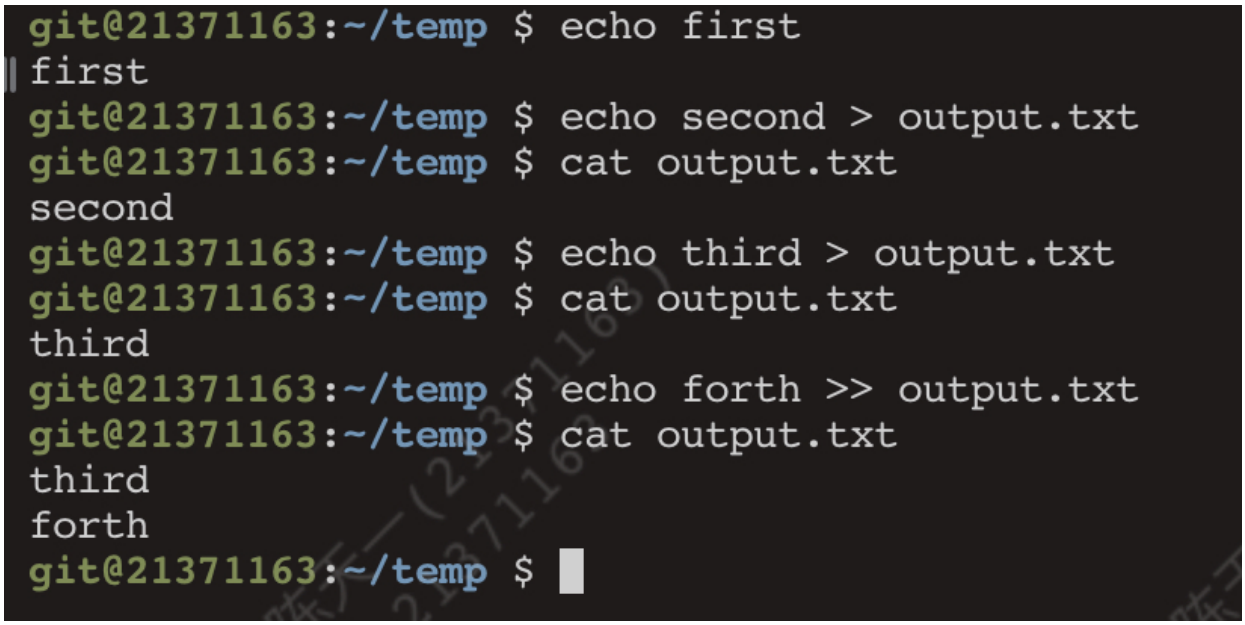
0.3

1. `git checkout -- print.c`
2. `git checkout HEAD print.c` then `git checkout -- print.c`
3. `git rm --cached hello.txt`

0.4

- 执行命令 `git reset --hard HEAD^` 后，提交日志只剩下1和2（及之前）的提交，没有了3的提交
- 执行命令 `git reset --hard <hash1>` 后，提交日志只剩下1的提交
- 执行命令 `git reset --hard <hash3>` 后，提交日志回到了有1、2、3号提交的状态

0.5



0.6

command 文件内容

```
echo "echo Shell Start..." > test
echo "echo set a = 1" >> test
echo "a=1" >> test
echo "echo set b = 2" >> test
echo "b=2" >> test
echo "echo set c = a+b" >> test
echo "c=$[$a+$b]" >> test
echo "echo c = $c" >> test
echo "echo save c to ./file1" >> test
echo "echo $c>file1" >> test
echo "echo save b to ./file2" >> test
echo "echo $b>file2" >> test
echo "echo save a to ./file3" >> test
echo "echo $a>file3" >> test
echo "echo save file1 file2 file3 to file4" >> test
echo "cat file1>file4" >> test
echo "cat file2>>file4" >> test
echo "cat file3>>file4" >> test
echo "echo save file4 to ./result" >> test
echo "cat file4>>result" >> test
```

result 文件内容

```
3
2
1
```

下为test的命令，解释见注释

```
```

```
echo Shell Start...
echo set a = 1
a=1 # a赋值1
echo set b = 2
b=2 # b赋值2
echo set c = a+b
c=$[$a+$b] # c赋值3
echo c = $c
echo save c to ./file1
echo $c>file1 # 向file1写入c的值3
echo save b to ./file2
echo $b>file2 # 向file2写入b的值2
echo save a to ./file3
echo $a>file3 # 向file3写入a的值1
echo save file1 file2 file3 to file4
cat file1>file4 # 向file4写入file1的内容3
cat file2>>file4 # 向file4追加file2的内容2，此时file4中有两行，分别为3和2
cat file3>>file4 # 向file4追加file3的内容1，此时file4中有三行，为3、2、1
echo save file4 to ./result
cat file4>>result # 将file4的内容写入result，故其与file4一致，有三行，为3、2、1
```

- `echo echo Shell Start` 与 `echo `echo Shell Start`` 有区别，前者输出 `echo Shell Start` 而后者输出 `Shell Start`
- `echo echo $c>file1` 与 `echo `echo $c>file1`` 效果有区别，前者在 file1 中写入 echo 字段和 c 的值，后者只写入 c 的值

难点分析

Lab0实际难度较小，主要难点在于Linux命令的使用、shell脚本的多种操作和gcc的用法。

遇到的主要困难有

- shell 脚本中管道的用法
- shell 脚本传参的方法
- shell 脚本控制语句的用法
- gcc 的编译与链接参数

实验体会

实验中实际所需操作量较小，但花费时间并不能算短，大部分时间都花费在查找命令和各种用法上，还是对相关的内容不够熟悉，需要更加仔细地阅读指导书。

但通过 Lab0 的实验，我认识到了 shell 脚本和 Makefile 等的快捷性与便利性，也让我对之后的实验产生了浓厚的兴趣。