lab0课下思考题

Thinking 0.1 git的使用1

不一样。

在README.txt新建的时候，其处于为未跟踪状态（untracked）；

在add并修改README.txt之后，其处于被修改状态（modified）。

Thinking 0.2 箭头与指令

- **add the file:** git add
- **stage the file:** git add
- **commit:** git commit

Thinking 0.3 小明的困境

- 从暂存区找回文件

```
git checkout printf.c
```

- 将最近一次commit版本中的该文件拉取到暂存区，之后再从暂存区找回该文件

```
git reset HEAD
git checkout printf.c
```

- 编辑.gitignore文件，将Tucao.txt添加进去，并删除暂存区的Tucao.txt

```
vim .gitignore  #之后进行添加
git rm Tucao.txt
```

Thinking 0.4 git的使用2

理解了reset还原版本的用法即可。

Thinking 0.5 克隆命令

- 错误，只会克隆 HEAD 指向的分支，第一次使用某一个分支需要用 git checkout 才能克隆。
- 正确
- 正确
- 正确，值得一提的是在一些社会运动之后，GitHub上的master分支重命名为了main分支。

Thinking 0.6 echo的使用

- **echo first:** 在命令行界面输出first
- **echo second > output.txt:** 向output.txt其中写入second，如果没有output.txt则新建output.txt，如果原本有output.txt且里面有内容，则覆盖原内容
- **echo third > output.txt:** 向output.txt其中写入third，如果没有output.txt则新建output.txt，如果原本有output.txt且里面有内容，则覆盖原内容
- **echo forth >> output.txt:** 向已有的output.txt中追加写入forth

Thinking 0.7 文件的操作

command文件的内容：

```
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文件的内容：

```
3
2
1
```

test文件中，定义了变量a、b、c并赋予了初值，再将其值分别写入file1、file2、file3中，再将3个新建的文件中的内容写入file4中，最后将file4中的内容写入result中。

echo Shell Start 与 echo ‘echo Shell Start’ 效果相同

echo echo $c>file 向 file 写入 echo $c echo 'echo $c>file1' 在终端打印 echo $c>file