# Lab 0 Report

21371295 张昊翔

## 思考题

### Thinking 0.1

Untracked.txt中内容：

```
git@21371295:~/learnGit (master)$ cat Untracked.txt
位于分支 master

尚无提交

未跟踪的文件:
  （使用 "git add <文件>..." 以包含要提交的内容）
        Untracked.txt
        readme.txt

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）
```

Stage.txt中内容：

```
git@21371295:~/learnGit (master)$ cat Stage.txt
位于分支 master

尚无提交

要提交的变更：
  （使用 "git rm --cached <文件>..." 以取消暂存）
        新文件：    readme.txt

未跟踪的文件:
  （使用 "git add <文件>..." 以包含要提交的内容）
        Stage.txt
        Untracked.txt
```

Modified.txt中内容：

```
git@21371295:~/learnGit (master)$ cat Modified.txt
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git restore <文件>..." 丢弃工作区的改动）
        修改：      readme.txt

未跟踪的文件:
  （使用 "git add <文件>..." 以包含要提交的内容）
        Modifie.txt
        Stage.txt
        Untracked.txt

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
```

不一样。未跟踪和已修改的区别是暂存区中有无该文件的历史记录。

## Thinking 0.2

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

## Thinking 0.3

1. git checkout -- print.c

2. git reset HEAD print.c; git checkout -- print.c
3. git rm --cached print.c

## Thinking 0.4

版本回退命令使用Head^：`git reset --hard HEAD^`
hash值是git log显示日志中的commit后字符串，使用hash值可以在任意版本之间进行切换：`git reset --hard <hash>`

## Thinking 0.5

```
git@21371295:~/learnGit (master)$ echo first
first

git@21371295:~/learnGit (master)$ echo second > output.txt
git@21371295:~/learnGit (master)$ cat output.txt
second

git@21371295:~/learnGit (master)$ echo third > output.txt
git@21371295:~/learnGit (master)$ cat output.txt
third

git@21371295:~/learnGit (master)$ echo forth >> output.txt
git@21371295:~/learnGit (master)$ cat output.txt
third
forth
```

>>为追加输出,否则覆盖原有内容

## Thinking 0.6

command中内容：

```
echo echo Shell Start...
echo echo set a = 1
echo a=1
echo echo set b = 2
echo b=2
echo echo set c = a+b
echo 'c=$[$a+$b]'
echo 'echo c = $c'

echo echo save c to ./file1
echo 'echo $c>file1'
echo echo save b to ./file2
echo 'echo $b>file2'
echo echo save c to ./file1
echo 'echo $a>file3'

echo echo save file1 file2 file3 to file4
echo 'cat file1>file4'
echo 'cat file2>>file4'
echo 'cat file3>>file4'
echo echo save file4 to ./result
echo 'cat file4>>result'
```

result中内容：

```
Shell Start...
set a = 1
set b = 2
set c = a+b
c = 3
save c to ./file1
save b to ./file2
save c to ./file1
save file1 file2 file3 to file4
save file4 to ./result
3
2
1
```

有区别，若含有反引号，会先执行反引号中的命令，并将执行结果作为原命令的输出。

- echo echo Shell Start: echo Shell Start
- echo `echo Shell Start`: shell start

## 难点分析

- 0.1：sed命令中选项的用法
- 0.2：shell中while循环与if的格式书写
- 0.3：grep命令输出的具体形式，其与awk命令的综合运用
- 0.4：Makefile的嵌套使用，多个command并行处理（解决方法：一行内用分号连接多个命令）

## 实验体会

当我第一次接触命令行界面(CLI)时，很不适应靠一条条命令完成基本操作的方法，但熟悉之后，初步体会到了它的简洁高效，期待后续的探索。与此同时，对多种文件中不同语法的学习让我在实验中效率很低，经常查阅资料，需要多练以熟能生巧。