北航操作系统实验报告 Lab 0

学号：71066001

姓名：陈伟杰

思考题

Thinking 0.1

Thinking 0.1 思考下列有关 Git 的问题：

- 在/home/21xxxxxx/learnGit（已初始化）目录下创建一个名为 README.txt 的文件。执行命令 git status > Untracked.txt。
- 在 README.txt 文件中添加任意文件内容，然后使用 add 命令，再执行命令 git status > Stage.txt。
- 提交 README.txt，并在提交说明里写入自己的学号。
- 执行命令 cat Untracked.txt 和 cat Stage.txt，对比两次运行的结果，体会 README.txt 两次所处位置的不同。
- 修改 README.txt 文件，再执行命令 git status > Modified.txt。
- 执行命令 cat Modified.txt，观察其结果和第一次执行 add 命令之前的 status 是否一样，并思考原因。

■

<Untracked.txt>

Untracked.txt

```
On branch Gitlearn
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

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

<Stage.txt>

Stage.txt

```
On branch Gitlearn
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.txt

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

<Modifield.txt>

Modifield.txt

```
On branch Gitlearn
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

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

< 编 写 过 程 >

```
Last login: Wed Mar  1 20:49:15 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Air:~ alex$ git status > Untracked.txt
warning: could not open directory '.Trash/': Operation not permitted
MacBook-Air:~ alex$ vim README.txt
MacBook-Air:~ alex$ git add README.txt
MacBook-Air:~ alex$ git status > Stage.txt
warning: could not open directory '.Trash/': Operation not permitted
MacBook-Air:~ alex$ git commit -m "71066001"
Auto packing the repository in background for optimum performance.
See "git help gc" for manual housekeeping.
warning: The last gc run reported the following. Please correct the root cause
and remove .git/gc.log.
Automatic cleanup will not be performed until the file is removed.

warning: There are too many unreachable loose objects; run 'git prune' to remove them.

[Gitlearn ad33d26] 71066001
 1 file changed, 1 insertion(+), 1 deletion(-)
MacBook-Air:~ alex$ cat Untracked.txt
On branch Gitlearn
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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        #include <stdio.h>.c
        .500729.padl
        .7d9b55297a162003a6d46e85fb151995daed6138
        .AntssX-NG/
        .CFUserTextEncoding
        .DS_Store
        .TranslationPlugin/
        .WestWorldSS/
        .adobe/
        .alex
        .bash_history
        .bash_profile
        .bash_profile.swi
        .bash_profile.swj
        .bash_profile.swk
        .bash_profile.swl
        .bash_profile.swm
        .bash_profile.swn
        .bash_profile.swo
        .bash_profile.swp
        .bash_sessions/
        .bytertc/
        .cache/
        .conda/
        .condarc
        .config/
        .cpan/
        .cups/
        .gitconfig
        .idlerc/
        .jssc/
        .keras/
        .kingsoft/
        .lesshst
```

```
MacBook-Air:~ alex$ vim README.txt
MacBook-Air:~ alex$ git status > Modifield.txt
warning: could not open directory '.Trash/': Operation not permitted
MacBook-Air:~ alex$ cat Modifield.txt
On branch Gitlearn
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

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

```
        .npmrc
        .nutstore/
        .oracle_jre_usage/
        .redhat/
        .rest-client/
        .ros/
        .ssh/
        .swt/
        .tcshrc
        .viminfo
        .vscode/
        .vue-templates/
        .xmlresolver.org/
        .xonshrc
        .zshrc
        Applications (Parallels)/
        Applications/
        Creative Cloud Files/
        Desktop/
        Documents/
        Downloads/
        IdeaProjects/
        Library/
        MIPS-Single.circ
        MIPS-Single外部库.circ
        Modifield.txt
        Movies/
        Music/
        Nutstore Files/
        NutstoreCloudBridge/
        OO/
        P3课下作业.circ
        PYTHON/
        Pictures/
        Sites/
        Stage.txt
        Untracked.txt
        VirtualBox VMs/
        aiXcoder/
        alu.v
        apache-maven-3.8.5/
        aurora/
        c-projects/
        file1
        file2
        file3
        file4
        homework_1/
        iCloud Drive（归档）/
        iCloud云盘（归档）/
        java排序.java
        os thinking总集/
        p0.circ
        package-lock.json
        package.json
        python练习/
        result
        test.py
        test/
        weijie.c/
        火神山.py
        练习java.java

no changes added to commit (use "git add" and/or "git commit -a")
MacBook-Air:~ alex$
```

没有执行 add 命令之前的文件状态不一样，因为 add 命令会将指定文本追加到原文件中，而 cat 命令只会查看现有文件的内容。

Thinking0.2

> **Thinking 0.2** 仔细看看0.10，思考一下箭头中的 add the file 、stage the file 和 commit 分别对应的是 Git 里的哪些命令呢？ ■

Add the file = git add
Stage the file = git add
Commit = git commit

Thinking0.3

> **Thinking 0.3** 思考下列问题：
>
> 1. 代码文件 `print.c` 被错误删除时，应当使用什么命令将其恢复？
> 2. 代码文件 `print.c` 被错误删除后，执行了 `git rm print.c` 命令，此时应当使用什么命令将其恢复？
> 3. 无关文件 `hello.txt` 已经被添加到暂存区时，如何在不删除此文件的前提下将其移出暂存区？
>
> ■

1. 代码文件 print.c 被错误删除时,则使用 git checkout -- print.c 命令将其恢复
2. 此时应当使用 git reset HEAD print.c 将其恢复
3. 在不删除 hello.txt 的前提下将其移出暂存区需要 git reset HEAD hello.txt

Thinking 0.4

Thinking 0.4 思考下列有关 Git 的问题:

- 找到在/home/21xxxxxx/learnGit 下刚刚创建的 README.txt 文件，若不存在则新建该文件。
- 在文件里加入 Testing 1，git add，git commit，提交说明记为 1。
- 模仿上述做法，把 1 分别改为 2 和 3，再提交两次。
- 使用 git log 命令查看提交日志，看是否已经有三次提交，记下提交说明为 3 的哈希值[a]。
- 进行版本回退。执行命令 git reset --hard HEAD^后，再执行 git log，观察其变化。
- 找到提交说明为 1 的哈希值，执行命令 git reset --hard <hash> 后，再执行 git log，观察其变化。
- 现在已经回到了旧版本,为了再次回到新版本,执行 git reset --hard <hash>，再执行 git log，观察其变化。

1.git reset --hard HEAD^后是回退到上一个版本，所以第三次提交日记没有了
2.git reset --hard 后面跟你要回退的那个版本的 commit 值，所以只保留到该哈希值的提交日志，那么第二次提交日志就会消失

Thinking0.5

Thinking 0.5 执行如下命令, 并查看结果

- `echo first`
- `echo second > output.txt`
- `echo third > output.txt`
- `echo forth >> output.txt`

```
Last login: Wed Mar  1 12:21:34 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Air:~ alex$ echo first
first
MacBook-Air:~ alex$ echo second > output.txt
MacBook-Air:~ alex$ echo third > output.txt
MacBook-Air:~ alex$ echo forth >> output.txt
MacBook-Air:~ alex$
```

output..txt — 已编辑

```
second
```

output.txt

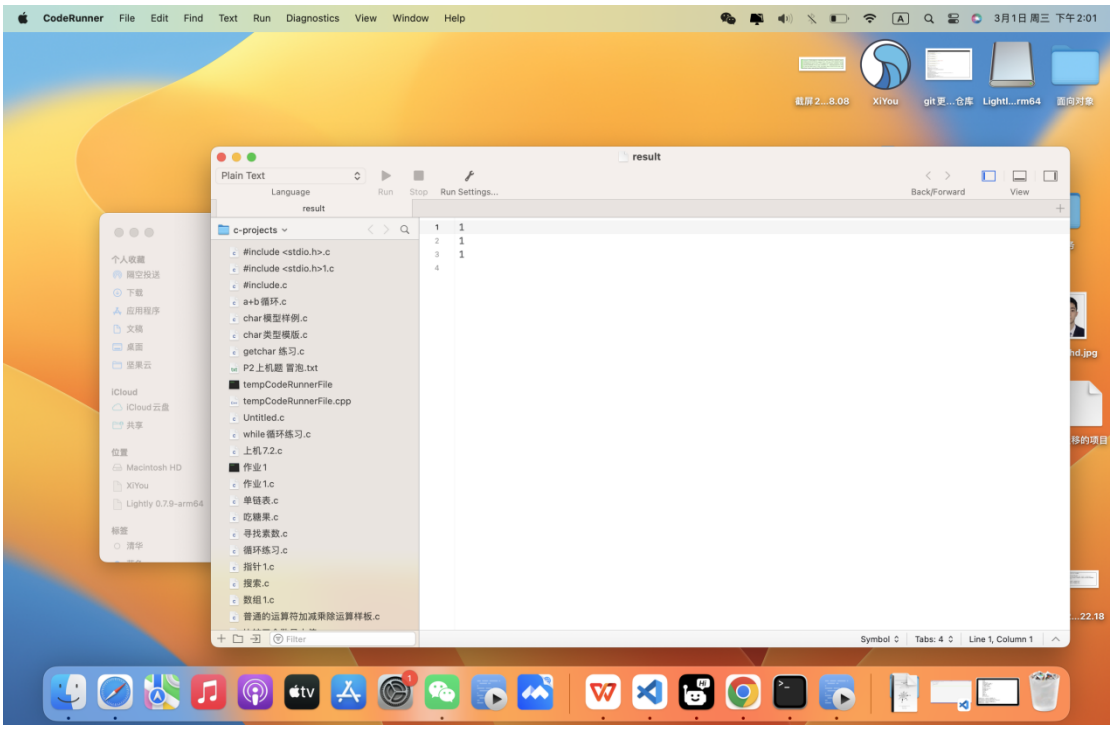
```
third
forth
```

Thinking0.6

> **Thinking 0.6** 使用你知道的方法（包括重定向）创建下图内容的文件（文件命名为 `test`），将创建该文件的命令序列保存在 `command` 文件中，并将 `test` 文件作为批处理文件运行，将运行结果输出至 `result` 文件中。给出 `command` 文件和 `result` 文件的内容，并对最后的结果进行解释说明（可以从 `test` 文件的内容入手）. 具体实现的过程中思考下列问题: `echo echo Shell Start` 与 `echo` `` `echo Shell Start` ``效果是否有区别; `echo echo $c>file1` 与 `echo` `` `echo $c>file1` ``效果是否有区别. ■

指导书给的 command 内容

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



定义了三个变量 a、b、c，a 赋值为 1，b 赋值为 2，c 赋值为 a+b 即为 3。将 c、b、a 分别存入 file1、file2、file3 中，再依此定向写入 file4 中，最后将 file4 中结果存入 result

echo echo Shell Start 与 echo `echo Shell Start`效果是否有区别?
答：是的，第一个命令将会输出两个“echo shell start”，而第二个命令则会解释`echo shell start`并执行它。

echo echo $c>file1 与 echo `echo $c>file1`效果是否有区别.

答：是的，第一个命令将会输出 echo $c>file1，而第二个命令则会解释`echo $c>file1`并将$c 的值写入 file1 文件中。

困难问题

由于是第一次接触 Linux 和 Git， vim 模式也是琢磨了好久，所以在做 lab0 的时候遇到了比较多困难，在 lab 0 反复编写 shell 脚本和 make 文件。

遇到的情况：

1. 不熟悉各种命令的使用，得一直看指导书和 os 平台的实验教程来看
2. makefile 的编写（了解 makefile 的基础规则)
3. sed 的使用在 hello_os.sh 极为重要，因为我可以用 sed 来提取相关行的内容
4. 在做实验的时候想要查看文件权限，如果权限不足就要 chmod+x test.sh

gcc 命令行（了解怎么从编译到可执行文件的过程及写法 参考指导书例子） -o , -c, -s 后都要紧跟文件名（路径），否则无法识别参数

```
all: hello_world.c
	gcc -o hello_world hello_world.c
clean:
	rm -f helloworld
```

5. git 的使用（更多的比如 commit 错了 或者提交我不想提交的那份要用到 git reset --hard 等）
6. 掌握 linux 的命令， 比如跳目录的命令 cd，新建文件的 touch 命令，rm 删除命令
7. make 的内外层调用
8. Vim 模式 比如 insert 模式 编译完成后按 esc 然后：wq 保存即可
9. 涉及管道、grep 的关知识
10. If 格式的认识
11. 变量名的命名规范

实验体会

在这次第一次实验中，由于之前没学过相关命令行及操作，所以在做的时候比较吃力，因为要一直看指导书之余，还要网上查找资料，归根到底还是自己没掌握好相关知识，但我觉得经过这次的铺垫和练习能为我日后在后续的 lab 打好基础，只有底子打好了后面才会游刃有余。