Lab0实验报告

一、思考题

1. Thinking 0.1

> Thinking 0.1 思考下列有关 Git 的问题：
> • 在前述已初始化的 ~/learnGit 目录下，创建一个名为 README.txt 的文件。执行命令 git status > Untracked.txt（其中的 > 为输出重定向，我们将在 0.6.3 中详细介绍）。
> • 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。

answer：

- 执行git status > Untracked.txt ，表示查询当前README.txt文件状态，并将其记录在Untracked.txt文件中；git status > Stage.txt和git status > Modified.txt 同理。
- cat Untracked.txt后，第二行显示Untracked files:，说明：在 README.txt 新建的时候，其处于为未跟踪状态 （untracked）；
- cat Stage.txt后，第二行显示 Changes to be committed:，说明：在 README.txt 中任意添加内容，接着用 add 命令之后，文件处于暂存状 态（staged）；
- cat Modified.txt后，第二行显示 Changes not staged for commit:，说明：在修改 README.txt 之后，其处于被修改状态（modified）。

2. Thinking 0.2

> 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？

answer

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

3. Thinking 0.3

> 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？

answer

- git checkout --print.c

- git reset HEAD print.c && git checkout --print.c
- git rm -- cached print.c

4. Thiking 0.4

> Thinking 0.4 思考下列有关 Git 的问题：
> • 找到在 /home/22xxxxxx/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。

```
git@23371331:~/learnGit (master)$ git log
commit ce1b316feeffe9228aca926dc0718596fd011510 (HEAD -> master)
Author: Liu <2936838784@qq.com>
Date:   Tue Mar 11 19:28:04 2025 +0800

    3

commit 96410c384eb3b1ce55b304fdc6e6345e9b85c4d3
Author: Liu <2936838784@qq.com>
Date:   Tue Mar 11 19:27:31 2025 +0800

    2

commit 03e031fc805c7c66abba80fec3a03f7062efa26e
Author: Liu <2936838784@qq.com>
Date:   Tue Mar 11 19:26:44 2025 +0800

    1

commit 5777b84026fe165f05574785c9418609fdb6457a
Author: Liu <2936838784@qq.com>
Date:   Sat Mar 1 18:56:08 2025 +0800

    23371331

commit 3116346a83c05340ec3f7a84da548870a9aa02c7
Author: 刘誉洲 <23371331@buaa.edu.cn>
Date:   Sat Mar 1 18:44:15 2025 +0800

    Notes to test
```

5. Thiking 0.5

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

answer

```
git@23371331:~/learnGit (master)$ echo first
first
```

```
git@23371331:~/learnGit (master)$ echo second > output.txt
git@23371331:~/learnGit (master)$ cat output.txt
second
```

```
git@23371331:~/learnGit (master)$ echo third > output.txt
git@23371331:~/learnGit (master)$ cat output.txt
third
```

```
git@23371331:~/learnGit (master)$ echo forth >> output.txt
git@23371331:~/learnGit (master)$ cat output.txt
third
forth
```

`echo`指令向标准输出中输出后续内容，使用`>`可把输出定向到符号后的文件（重写）；使用`>>`符号可把输出**追加**到目标文件中

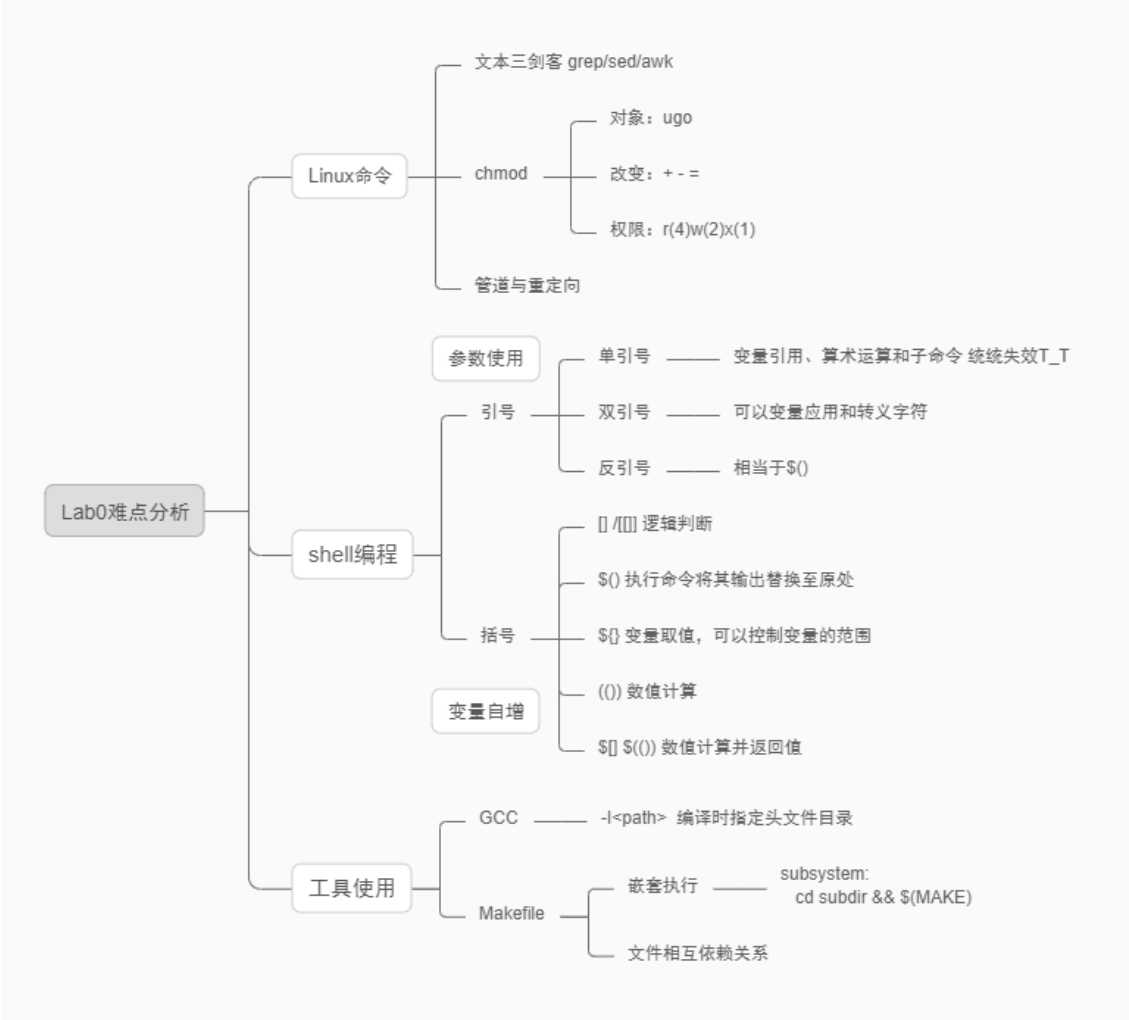
6. Thinking 0.6

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.

```
git@23371331:~/learnGit (master)$ vim command
git@23371331:~/learnGit (master)$ chmod +x command
git@23371331:~/learnGit (master)$ bash command
git@23371331:~/learnGit (master)$ ls
command  test
git@23371331:~/learnGit (master)$ cat test
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
```

```
#!/bin/bash                                                    touch test
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
```

难点分析：

- 

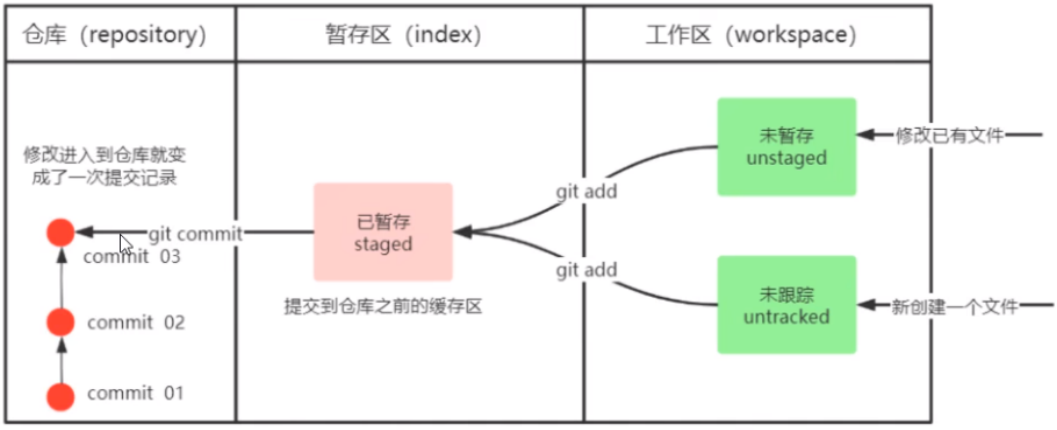
- 使用let时等号右边的值若小于零则退出状态码返回1，当然对代码本身执行没有影响，但必须换成算数扩展，例如 n=$((n-1))；也可以暴力覆盖退出码，适用于需要保留 `let` 的场景，用exit 0。
- awk的用法介绍：

```
awk '$1>2 {print $1,$3}' my.txt
$n 代表每一行中用空格分隔后的第n项
意义是输出文件 my.txt 中所有第一项大于 2 的行的第一项和第三项

awk -F, '{print $2}' my.txt
-F 选项用来指定用于分隔的字符，默认是空格
该命令的 $n 就是用“,”分隔的第 n 项
```

- `Makefile`调用子目录下的`Makefile`

```
$(MAKE) -C subdir
```

- 

  熟悉git add / git commit包括删除文件的恢复等操作
- 当变量名与其他字符相邻时，必须用 `{}` 明确变量范围，否则 Shell 会误解变量名。
- **错误示例**

  bash

```
n=5
echo "第$n个苹果"     # 正确写法：第5个苹果
echo "价格$n美元"     # 正确写法：价格5美元
echo "总价$np美元"    # 错误！Shell 会将 `$np` 视为变量名（但未定义）
```

  正确写法

  bash

```
echo "总价${n}p美元"  # 明确变量名边界 → 总价5p美元
```

  - eq代表算数相等，而==用于字符串比较
  - chmod修改文件权限为"r--r-----"应该

实验体会

1. lab0这次后面的题没做出来，归根结底是课下学习不细心导致的。一方面，chmod修改文件权限没有很好的理解，只停留在基础层面，遇到"r--r-----"就搞不懂是什么意思了，其实在视频和指导书里都有解释到。还有awk等指令不熟悉，导致还要一个个尝试那段写法正确浪费时间。
2. 考试的时候心态过于紧张也是需要改进的点，当时chmod没搞懂明明可以多翻翻指导书的，下次上机要调整好心态

原创说明

1. 难点分析的图1，图2来自https://yanna-zy.github.io/