操作系统第一周作业

PART 1 实验思考题

练习 0.1

在 bash 中分别输入

• echo "Hello Ubuntu"

• bash –version

• ls

三条命令，简单思考其回显结果

A: 1. 显示 Hello Ubuntu，echo 命令能将后面跟着的语句输出一次

2. 查看 bash 版本

3. 列出当前目录下的所有文件

思考题 0.1

Q: 通过你的使用经验，简单分析 CLI Shell，GUI Shell 在你使用过程中的各自优劣（100 字以内）

A: GUI shell 即图形界面。

优点：

1. 直观清晰、美观、人性化。
2. 易上手。

缺点：

1. 不好的设计反而更难用。
2. 复杂的软件其图形界面很难设计出色。

CLI 即命令行界面。

优点：

1. 工作效率高。
2. 高性能。
3. 按指令执行更加精确，避免软件 bug。

缺点：

1. 文本方式工作，界面不美观。
2. 需查阅文档，不易上手。

思考题 0.2

Q: 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）

A: command 文件内容：

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

result 文件内容：

3
2
1

解释说明：

1. test 文件是批处理文件，在批处理中的命令可以通过命令行窗口执行。这道题中包括 echo 和 cat 以及重定向等命令。
2. echo 这个命令能将后面跟着的语句输出一次。
3. 重定向符>将输出重定向到文件中。>>以追加的方式将内容重定向到文件。
4. $是批处理脚本中变量的存储方式。在使用已经定义过的变量时，在变量前面增加美元符号$
5. cat 命令用于连接文件并打印到标准输出设备上。用重定向符可以将内容重定向到文件中。
6. 运算符。语法是：$((表达式))。

练习 0.3

- *在/home/17xxxxxx_2019_jac/learnGit（已 init）目录下创建一个名为 README.txt 的文件。这时使用 git status > Untracked.txt 。*
- *在 README.txt 文件中随便写点什么，然后使用刚刚学到的 add 命令，再使用 git status > Stage.txt 。*

- *之后使用上面学到的 Git 提交有关的知识把 README.txt 提交，并在提交说明里写入自己的学号。*
- *使用 cat Untracked.txt 和 cat Stage.txt，对比一下两次的结果，体会一下 README.txt 两次所处位置的不同。*
- *修改 README.txt 文件，再使用 git status > Modified.txt 。*
- *使用 cat Modified.txt ，观察它和第一次 add 之前的 status 一样吗，思考一 下为什么？*

A: 当没有 add 时，本地创建或更改的文件没有被 track。add 之后，文件放入暂存区。当 README.txt 被修改时，在 add 之前，更新内容没有被放入暂存区。

思考题 0.3

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？

A: add the file: git add

stage the file: git add

commit: git commit

思考题 0.4

- *深夜，小明在做操作系统实验。困意一阵阵袭来，小明睡倒在了键盘上。等到小明早上醒来的时候，他惊恐地发现，他把一个重要的代码文件 printf.c 删除掉了。苦恼的小明向你求助，你该怎样帮他把代码文件恢复呢？*
- *正在小明苦恼的时候，小红主动请缨帮小明解决问题。小红很爽快地在键盘上敲下了 git rm printf.c，这下事情更复杂了，现在你又该如何处理才能弥补小红的过错呢？*
- *处理完代码文件，你正打算去找小明说他的文件已经恢复了，但突然发现小明的仓库里有一个叫 Tucao.txt，你好奇地打开一看，发现是吐槽操作系统实验的，且该文件已经被添加到暂存区了，面对这样的情况，你该如何设置才能使 Tucao.txt 在不从工作区删除的情况下不会被 git commit 指令提交到版本库*

A: 1. git checkout -- printf.c

2. git log 查看版本库信息

git checkout id

git reset --hard id # 可恢复到版本 id

3. git rm --cached Tucao.txt，从暂存区删除文件，而工作区不受影响

练习 0.4

- *找到我们在/home/17xxxxxx_2019_jac/下刚刚创建的 README.txt，没有的话就新建一个。*
- *在文件里加入 Testing 1，add，commit，提交说明写 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 命令查看一下提交日志，看是否已经有三次提交了？记下提交说明为 3 的哈希值 1。*

- *开动时光机！使用 git reset --hard HEAD^ ，现在再使用 git log，看看什么没了？*
- *找到提交说明为 1 的哈希值，使用 git reset --hard <Hash-code> ，再使用 git log，看看什么没了？*
- *现在我们已经回到过去了，为了再次回到未来，使用 git reset --hard <Hash-code> ，再使用 git log，我胡汉三又回来了！*

A: 使用 git reset --hard HEAD^, commit message 为 3 的一次 commit 没了
git reset --hard <Hash-code1>，回到了 commit 内容为 1 的一次 commit，commit 为 2 的没了

思考题 0.5

思考下面四个描述，你觉得哪些正确，哪些错误，请给出你参考的资料或实验证据。

- *克隆时所有分支均被克隆，但只有 HEAD 指向的分支被检出。*
- *克隆出的工作区中执行 git log、git status、git checkout、git commit 等操作不会去访问远程版本库。*
- *克隆时只有远程版本库 HEAD 指向的分支被克隆。*
- *克隆后工作区的默认分支处于 master 分支。*

A: 1.错误，克隆时仅 HEAD 所指向的当前分支被克隆。

```
-bash-3.2$ git branch -a
* master
  remotes/origin/master
```

2. git log 呈现 commit 情况；git checkout 切换分支，或者将 master 分支中的指定文件替换暂存区和以及工作区中的文件，因此会访问远程版本库。但 git status 呈现的是工作区和暂存区的情况，因此不会访问远程版本库。

3. 正确

4. 正确。从远程仓库克隆时，实际上 Git 自动把本地的 master 分支和远程的 master 分支对应起来了，并且，远程仓库的默认名称是 origin。

练习 0.5

仔细回顾一下上面这些指令，然后完成下面的任务

- *在 /home/17xxxxxx_2019_jac/17xxxxxx-lab 下新建分支，名字为 Test*
- *切换到 Test 分支，添加一份 readme.txt，内容写入自己的学号*
- *将文件提交到本地版本库，然后建立相应的远程分支。*

A：创建分支，切换分支，新建 readme.txt，并写入学号

```
17373123_2019_jac@stu-113:~/17373123-lab$ git branch Test
17373123_2019_jac@stu-113:~/17373123-lab$ git checkout Test
 Switched to branch 'Test'
17373123_2019_jac@stu-113:~/17373123-lab$ touch readme.txt
17373123_2019_jac@stu-113:~/17373123-lab$ echo 17373123 >> readme.txt
```

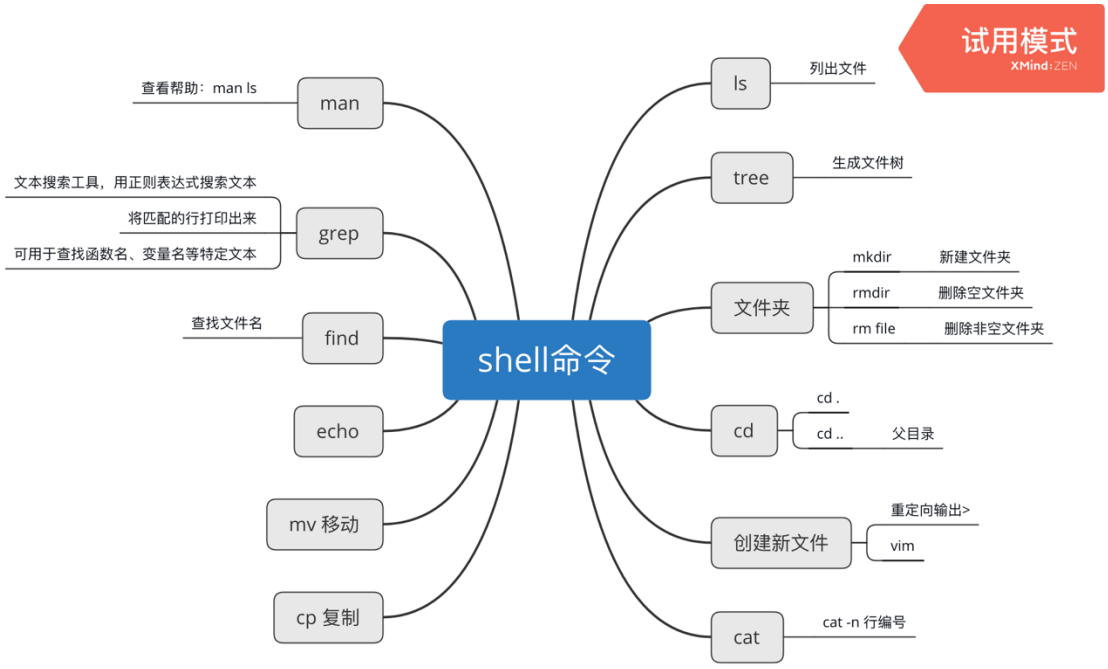
add, commit, push

```
[17373123_2019_jac@stu-113:~/17373123-lab$ git add readme.txt
[17373123_2019_jac@stu-113:~/17373123-lab$ git status
 # On branch Test
 # Changes to be committed:
 #   (use "git reset HEAD <file>..." to unstage)
 #
 #       new file:   readme.txt
 #
[17373123_2019_jac@stu-113:~/17373123-lab$ git commit -m "Test"
 [Test 4f8a6e8] Test
  1 file changed, 1 insertion(+)
  create mode 100644 readme.txt
```

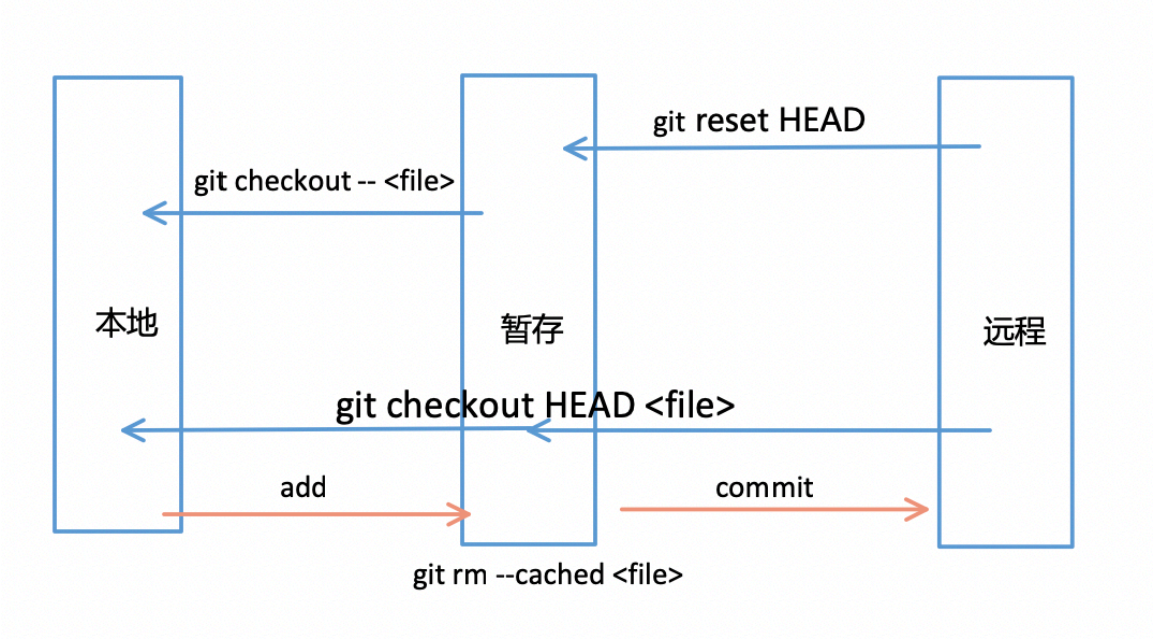
PART 2　课下实验难点与体会

难点：

1. shell 命令的理解与使用



2. git 三棵树的理解，主要是从远程端往下比较难记



体会：

1. Makefile，gcc，vim，shell 命令，git，很多东西都是新的，一下子消化不了。
2. lab0 课下难度不大，但花了很长时间入门。时间大约四五个小时。尚未完全学会。
3. 网上查不到或者查到了还是没看懂，很难受。

PART 3　指导书反馈

1. cscore 清晰明白，很好。
2. 希望除了实验指导书之外能有可供查阅的手册或好的参考文档的链接推荐，网上资料泥沙俱下。
3. 课下作业要求希望能更清晰。刚接触对于这些东西尚不了解，一下看不懂题目要求做什么。比如，lab0 的其中一个实验：写一个 Makefile 并用它生成可执行文件，而不是直接输入 gcc 命令生成。做的时候有同学理解成只要生成了可执行文件就行。
4. 残留难点：Makefile，我觉得我缺乏实战练习。