Lab0 实验报告

19376273 陈厚伦

一、实验思考题

（一）思考题 0.1

CLI Shell:

优点：图形化界面直观，命令以按钮形式出现，可以让用户非常快的熟悉 git 命令，而且工作区暂存区显示出来，方便随时观察动态。

缺点：不够简洁，不能够显示所有命令的功能

GUI Shell:

优点：通过输入命令执行，非常准确

缺点：对新手用户不友好，学习成本高

（二）思考题 0.2

echo Shell Start 与 echo ‘Shell Start’没有区别，都是字符串 Shell Start
echo \$c>file1 与 echo ‘\$c>file1’有区别,第一个是把$c 字符串重定向写入 file1，第二个是直接标准输出字符串’$c>file1’

要注意$;* (空格)等特殊符号要进行转义
要注意第一次使用>，之后要注意是追加>>
command 文件：

```
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 “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：
```
    3
    2
    1
```
a=1,b=2,c=3,把 c 的值写入 file1，把 b 的值写入 file2，把 a 的值写入 file3，然后把 file1~3 依次重定向到 file4，再把 file4 重定向的 result。由于 file1~3 向 file4 写有追加操作，所以每次在文件末尾写，会有三行。result 中追加写入 file4 的内容，result 原有的内容不会被覆盖。

（三）思考题 0.3

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

（四）思考题 0.4

- 可以使用命令 git checkout -- printf.c
- 可以使用命令 git checkout -- printf.c
- 可以使用命令 git rm -- cached Tucao.txt 可以从缓存区中删除 Tucao.txt

（五）思考题 0.5

- 判断正误
  - 克隆时所有分支均被克隆，但只有 HEAD 指向的分支被检出。

  错误。不会克隆所有分支，只能克隆远程库的 master 分支。如果想要检出特定的分支还需要使用指令”git checkout 分支名”
  - 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。

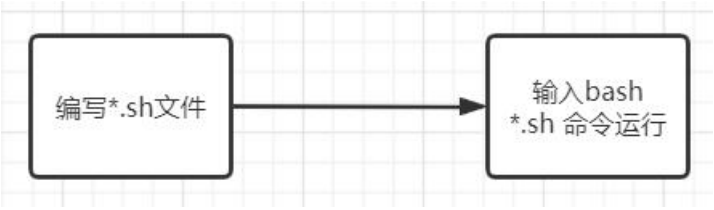
  正确。不会访问远程。上述命令都是本地命令，不会访问远程版本库。
  - 克隆时只有远程版本库 HEAD 指向的分支被克隆。

  正确。
  - 克隆后工作区的默认分支处于 master 分支。

  正确。

二、实验难点图示

（一）Shell 脚本编写与执行



注意事项：

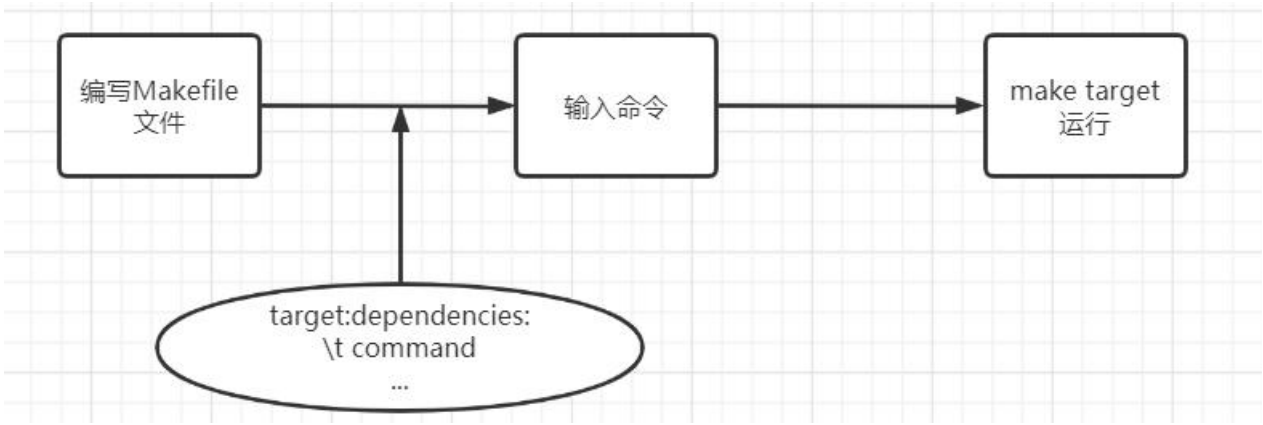
①表示字符串要使用$，传递参数按照顺序分别是$1,$2,$3...所有参数用$0

②变量与运算符之间要有空格，如 while [$a -ne 10]

③使用变量要用$，变量赋值时不要有空格，如 a=$a[$a+1]，这一点与高级语言编程风格不同，要格外注意

④文件最后结束时要多敲一个空行，作为正确的 EOF

（二）Makefile 的编写与调用



注意事项：

①Makefile 可以调用目录下的子目录中的 Makefile 文件，使用指令 cd ./xxx $$ make target

②make 单独使用时默认执行第一个 target，往往会把它设置成 all，作为执行的起点

③输入指令前要使用 tab，文件结尾多打一行空行

（三）重定向与管道

><重定向可以到某一个文件，也可以是某一个命令的输入输出

|后面要接命令

（四）gcc 编译工具

编译成.o 文件: gcc -c *.c

编译成.o 文件并链接生成可执行文件:gcc *.c -o *

先单独编译成.o 文件再执行链接会增加灵活性

三、体会与感想

本次实验实际难度不是很大，但是自己对于工具链不熟悉，导致许多情景下不清楚如何查资料，查什么样的资料，另外细节问题许多通过反复试错才慢慢领会。查阅资料并完成实验总体上花费 7-8h，还需要进一步提升自己使用工具链工具的能力，尽快适应这个“黑黑的命令窗口”。

四、指导书反馈

①建议专门开辟一个思考题栏目，把每一次实验的思考题汇总到这个栏目，方便作答。

②希望代码字体可以统一，使用代码风格字体，例如本报告只用的 Courier New 字体。

③建议引入部分从 CLI 开始过渡到 GUI，我 CLI 也没有接触过，直接用 GUI 有一定困难，所以有一个 CLI 来熟悉 git 再过渡到 GUI 会更好。

五、残留难点

对 Makefile 中 dependencies 难以理解原理，gcc 用的不熟练，Shell 脚本中 while、if 等控制流命令使用频度不够。