张金源 76066001

LAB0 实验报告

1. 思考题：

思考 0.1

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

GUI Shell 优点：

- Graphical User Interface – is a type of user interface that allows users to interact with electronic devices using images rather than text commands.
- 使用的时候很方便，让很多人很容易使用电脑。GUI 也让我们系统看起来更好看，优美，对刚开学使用电脑的人 GUI 很有帮助。GUI 的功能也很简单而丰富多彩。

GUI Shell 缺点：

- 虽然让我们系统看起来更好看，但一般 GUI 使用 RAM 比较多。比如如果你在 RAM 2GB 的电脑安装 Win 10, 另一个安装 Ubuntu Linux, 从我经验来说跑 Linux 的电脑不卡，但 Win10 的电脑很卡。The disadvantages of Graphical User Interface are it needs more memory of your computer. It is also difficult when it is not properly installed the program on your computer. GUI 有时候会让使用者的权利不够，因为有些 shell 的功能，如果用 CLI shell 才能实现。

CLI Shell 优点：

- Command Line Interface – is a means of interaction with a computer program where the user issues commands to the program in the form of successive lines of text.
- The advantages of Command Line Interface are easy to integrate with scripting and other programmatic scripts. It only uses a lower memory of the computer. It is also very accurate in piping multiple commands together。CLI 没有那么大使用 RAM，它也是系统最基本的 Shell，对电脑高手使用 CLI 有好多好出比如在 windows 我们可以使用 CMD 删除病毒，他的功能比 GUI 强很多，它查找一个文件也会很快，对我来说如果我们熟悉了 CLI Shell 我们会更了解我们的系统。

CLI Shell 缺点：

- The disadvantages of Command Line Interface are boring to look at. It also has no media. You cannot multitask. It requires hard work because you only use keyboard instead of using mouse through clicking. It is really really hassle for me. CLI Shell 对刚开学使用电脑的人会让他很

张金源 76066001

LAB0 实验报告

糊涂，因为 CLI 的命令很多，而它也不太容易使用。我个人学了 3 个月多才感觉比较熟悉了，CLI 其实也没有界面，这个对一般人会让他们很烦，而不是 User Friendly 的。

思考题 0.2

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）

1. 先创建一个文件用 vim, 命令为“vim test.txt”
2. 在 test.txt 里面插入要进行的命令

```
echo shell start…                    //print shell start
echo set a = 1                       // print set a =1
a=1                                  //取 a 值为 1
echo set b = 2                       //print set b =1
b=2                                  //取 b 值为 2
echo set c = a+b                     //print set c = a+b
c=$[$a+$b]                           //取 c 值为 a+b
echo c = $c                          //print c = $c 的值（a+b）
echo save c to ./file1               // print save c to ./file1
echo $c>file1                        // 插入 c 的值 to file1
echo save b to ./file2               // print save b to ./file2
echo $b>file2                        // 插入 b 的值 to file2
echo save a to ./file3               // print save a to ./file3
echo $a>file3                        //插入 a 的值 to file3
echo save file1 file2 file3 to file4 // print
cat file1>file4                      //插入 file1 to file4
cat file2>>file4                     //插入 file2 to file4 的下一行
```

张金源 76066001

LAB0 实验报告

cat file3>>file4　　//插入 file3 to file4 的下一行

echo save file4 to ./result　　// print save file4 to ./result

cat file4>>result　　//插入 file4 to result 的下一行

3. 然后保存，退出 vim 之后我们就可以运行 test.txt 里面的命令。我们使用 “source test.txt”，结果我们可以在 result 文件里面看见的。

思考题 0.3

仔细看看这张图，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/stage the file：$git add filename
commit: $git commit -m “comment of this 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？

git checkout -- <file> 例如：git checkout printf.c

•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？

git reset HEAD <file>　　例如：git reset HEAD printf.c

•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？

git clean <file> -f　　例如：git clean Tucao.txt -f

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。

1.**克隆时所有分支均被克隆，但只有 HEAD 指向的分支被检出。正确**

张金源 76066001

LAB0 实验报告

https://git-scm.com/docs/git-clone
https://blog.csdn.net/wh_19910525/article/details/7488931

2.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。错误

```
jovyan@c472baf150a5:~/work/76066001-lab$ git log
commit 3ab6766b31a2eddd97cfb3fead6bc3e533e3405a (HEAD -> lab0, origin/lab0)
Author: Michael <michael.201096@gmail.com>
Date:   Mon Mar 9 04:02:02 2020 +0000

    step 8 done

commit 7d468f0af5144e850ab7ea5f680bc8a614dadb4e
Author: Michael <michael.201096@gmail.com>
Date:   Mon Mar 9 01:57:10 2020 +0000

    step 7-fix

commit 845e80e585b7ac5c8487e29c5fbd7d42381eb5d1
Author: Michael <michael.201096@gmail.com>
Date:   Mon Mar 9 01:00:13 2020 +0000

    step 7

commit 548c6917547efeef74ec6115ed5168410548388b
Author: Michael <michael.201096@gmail.com>
Date:   Sun Mar 8 04:14:28 2020 +0000

    Step 6 done-fix

commit e1378cd6ca5398e0836b34cf3ef3267d73209628
Author: Michael <michael.201096@gmail.com>
Date:   Sun Mar 8 04:12:01 2020 +0000

    Step 6 done

commit 88e2e6d876aa3b585a08bffd3d0d0372f64a7182
Author: Michael <michael.201096@gmail.com>
Date:   Sun Mar 8 02:29:11 2020 +0000

    add dst

commit e9a67ac2a45bfb42b67e2e8719a76fb092bf991b
Author: buaa g306 <you@example.com>
Date:   Mon Mar 2 14:00:18 2020 +0800

    init lab0
```

```
jovyan@c472baf150a5:~/work/76066001-lab$ git commit
On branch lab0
Your branch is up to date with 'origin/lab0'.

nothing to commit, working tree clean
jovyan@c472baf150a5:~/work/76066001-lab$ git status
On branch lab0
Your branch is up to date with 'origin/lab0'.

nothing to commit, working tree clean
```

3.克隆时只有远程版本库 **HEAD** 指向的分支被克隆。 错误

Clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository (visible using `git branch -r`), and creates and checks out an initial branch that is forked from the cloned repository's currently active branch.

张金源 76066001

LAB0 实验报告

https://git-scm.com/docs/git-clone

4.**克隆后工作区的默认分支处于 master 分支。**正确

http://cscore.net.cn/courses/course-v1:Internal+B3I062140+2019_T2/courseware/153317b39af949f8b9df1c555d922732/cee76ce058174e4e8e1889e6bd7a233b/

我们克隆下来时默认处于 master 分支，但很可惜实验的代码是不会在 master 分支上测试的，所以我们要先使用检出对应的 labx 分支，再进行测试。

2. 实验难点图示

本实验的难点是在做第 8 步骤，在生成 Makefile 的时候遇到了各种各样的错误，如：当时想在 csc/code 导入 include 文件夹结果报错，然后想了半天才想到了一个办法就是在 csc/code 的 Makefile 只做 make clean,其他的在 csc 里面生成的 Makefile，在此点才能把 include 文件夹导入 Makefile 如下图所示：

```
jovyan@c472baf150a5:~/work/76066001-lab/csc/code$ cat Makefile
srcs:= fibo.c main.c
#cc:= g++ -wall -O2
#$headers:=~/work/76066001-lab/csc/include

#fibo.o: fibo.c $(headers)
#       gcc -I~/work/76066001-lab/csc/include/ -c fibo.c -o fibo.o

#main.o: main.c $(headers)
#       gcc -I$(headers) -c main.c -o main.o

clean:
        rm -f *.o
jovyan@c472baf150a5:~/work/76066001-lab/csc$ cat Makefile
dirs:=code
header:=include
modules := code
cc:= gcc -Wall -O2
srcs:=$(dirs)/fibo.c    \
        $(dirs)/main.c
headers:=$(header)/fibo.h
objects:= $(dirs)/main.o \
                $(dirs)/fibo.o

all: $(srcs) $(headers)
        gcc -I./include -c $(dirs)/fibo.c -o $(dirs)/fibo.o
        gcc -I./include -c $(dirs)/main.c -o $(dirs)/main.o
        gcc $(dirs)/main.o $(dirs)/fibo.o -o fibo

clean:
        $(MAKE) --directory=$(modules) clean;
        #for d in $(modules);    \
        #       do              \
                #       $(MAKE) --directory=$$d clean;  \
                #done;
```

张金源 76066001

LAB0 实验报告

3. 实验结果

```
jovyan@c472baf150a5:~/work/76066001-lab/src$ ls
Makefile  palindrome  palindrome.c  sh_test
jovyan@c472baf150a5:~/work/76066001-lab/src$ cat palindrome.c
#include<stdio.h>
int main()
{
        int n, reversedN = 0, remainder, originalN;
        scanf("%d",&n);
        originalN = n;

        while(n != 0){
                remainder = n%10;
                reversedN = reversedN*10+remainder;
                n /= 10;
        }









        if(originalN == reversedN){
                printf("Y");
        }else{
                printf("N");
        }
        return 0;
}
jovyan@c472baf150a5:~/work/76066001-lab/src$ cat Makefile
palindrome.exe:
        gcc palindrome.c -o palindrome
jovyan@c472baf150a5:~/work/76066001-lab/src$ cd sh_test
jovyan@c472baf150a5:~/work/76066001-lab/src/sh_test$ cat hello_os.sh
#!/bin/bash
#bshcopy

sed -n -e '8p;32p;128p;512p;1024p' $1 > $2
#bash hello_os.sh file hello_os.c
#$1 = file, $2 = hello_os.c
```

```
jovyan@c472baf150a5:~/work/76066001-lab/ray/sh_test1$ cat changefile.sh
#!/bin/bash
a=1
while [ $a -le 100 ]
do
        if [ $a -gt 70 ]           #if loop variable is greater than 70
        then
        rm -r file$a
        elif [ $a -gt 40 ]         # else if loop variable is great than 40
        then
        mv file$a newfile$a
        fi
        a=$(($a+1))                #don't forget change the loop variable
done
```

```
jovyan@c472baf150a5:~/work/76066001-lab/ray/sh_test2$ cat search.sh
#!bin/bash
#First you can use grep (-n) to find the number of lines of string.
#Then you can use awk to separate the answer.
word=$2
src=$1
dst=$3
grep -n $word $src | awk -F: '{print $1}' > $dst

#bash search.sh file int result
# $1 = file, $2 = int,  $3 = result
```

张金源 76066001

LAB0 实验报告

```
jovyan@c472baf150a5:~/work/76066001-lab/csc/code$ cat modify.sh
#!/bin/bash
#x=$2
#y=$3
sed -i -e "s/$2/$3/g" $1
#bash modify.sh fibo.c char int
#$1 = fibo.c(src), $2= char(search), #3= int(replace)
```

```
jovyan@c472baf150a5:~/work/76066001-lab/csc/code$ cat Makefile
srcs:= fibo.c main.c
#cc:= g++ -wall -O2
#$headers:=~/work/76066001-lab/csc/include

#fibo.o: fibo.c $(headers)
#       gcc -I~/work/76066001-lab/csc/include/ -c fibo.c -o fibo.o

#main.o: main.c $(headers)
#       gcc -I$(headers) -c main.c -o main.o

clean:
        rm -f *.o
```

```
jovyan@c472baf150a5:~/work/76066001-lab/csc$ cat Makefile
dirs:=code
header:=include
modules := code
cc:= gcc -Wall -O2
srcs:=$(dirs)/fibo.c    \
        $(dirs)/main.c
headers:=$(header)/fibo.h
objects:= $(dirs)/main.o \
                $(dirs)/fibo.o

all: $(srcs) $(headers)
        gcc -I./include -c $(dirs)/fibo.c -o $(dirs)/fibo.o
        gcc -I./include -c $(dirs)/main.c -o $(dirs)/main.o
        gcc $(dirs)/main.o $(dirs)/fibo.o -o fibo

clean:
        $(MAKE) --directory=$(modules) clean;
        #for d in $(modules);   \
        #       do              \
                #       $(MAKE) --directory=$$d clean;  \
                #done;
```

4. 体会与感想

在本次实验比较难的地方就是在做 Makefile 和做 modify.sh 这因为在做 Makefile 时需要想怎么构造 Makefile 然后能编译 fibo.c, main.c 和导入 fibo.h,还有生成他们的.o 文件。做 modify 时我遇到的困难就是比较模糊要使用 sed 或 awk，然后最后使用 sed 指令，但遇到另外个问题就是指令执行的时候在屏幕打印出输出，后来就加一个 parameter -i 表示“在当前文件执行”于是没有输出到屏幕。在本次实验我花了 8 个小时，因为有时候就想不到要怎么做，然后在第八步骤的时候花了比较多时间。我此实验体会感觉比较模糊，因为使用了新的平台，然后实验步骤比较多，感觉课下实验比去年更深，因为此实验有让我们掌握生成 Makefile，然后脚本的基本指令。

张金源 76066001

LAB0 实验报告

5. 残留难点

在本次实验因为我们使用了远控平台，而且这实验平台还是新的，我个人第一次上手的时候还不太熟悉，然后发现咱们的实验平台有点卡，就是每次要打指令有 delay 的，然后有时候会卡，需要 refresh。