Lab0实验报告

一、思考题

Thinking 0.1

> - cat Untracked.txt 显示 README.txt 未跟踪
> - cat Stage.txt 显示 README.txt 处于暂存区
> - cat Modified.txt 显示 README.txt 已修改但未暂存
>
> 不一样，因为文件被跟踪并修改了

Thinking 0.2

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

Thinking 0.3

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?

> git restore print.c

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?

> git reset HEAD print.c
>
> git restore print.c

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?

> git rm -- cached hello.txt

Thinking 0.4

> 使用这条命令可以进行版本回退或者切换到任何一个版本。它有两种用法：
>
> - 第一种是使用 HEAD 类似形式，如果想退回上个版本就用 HEAD^，上上个版本的话就用
>
> HEAD^^，要是回退到前 50 个版本则可使用 HEAD~50 来代替；
>
> - 第二种就是使用 hash值，使用 hash 值可以在不同版本之间任意切换

Thinking 0.5

> - echo first
>
> 在终端输出 `first`
>
> - echo second > output.txt
>
> 向 output.txt（不存在则创建）中写入 `second`

- echo third > output.txt

向 output.txt 中覆盖式写入 `third`

- echo fourth >> output.txt

向 output.txt 中追加式写入 `third`

Thinking 0.6

- command 文件

```
#!/bin/bash
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
```

- result 文件

```
3
2
1
```

`echo echo Shell Start`与`echo 'echo Shell Start'`没有区别

`echo echo $c>file1`与`echo 'echo $c>file1'`有区别

前者会将 "echo $c" 重定向输出到 file1，

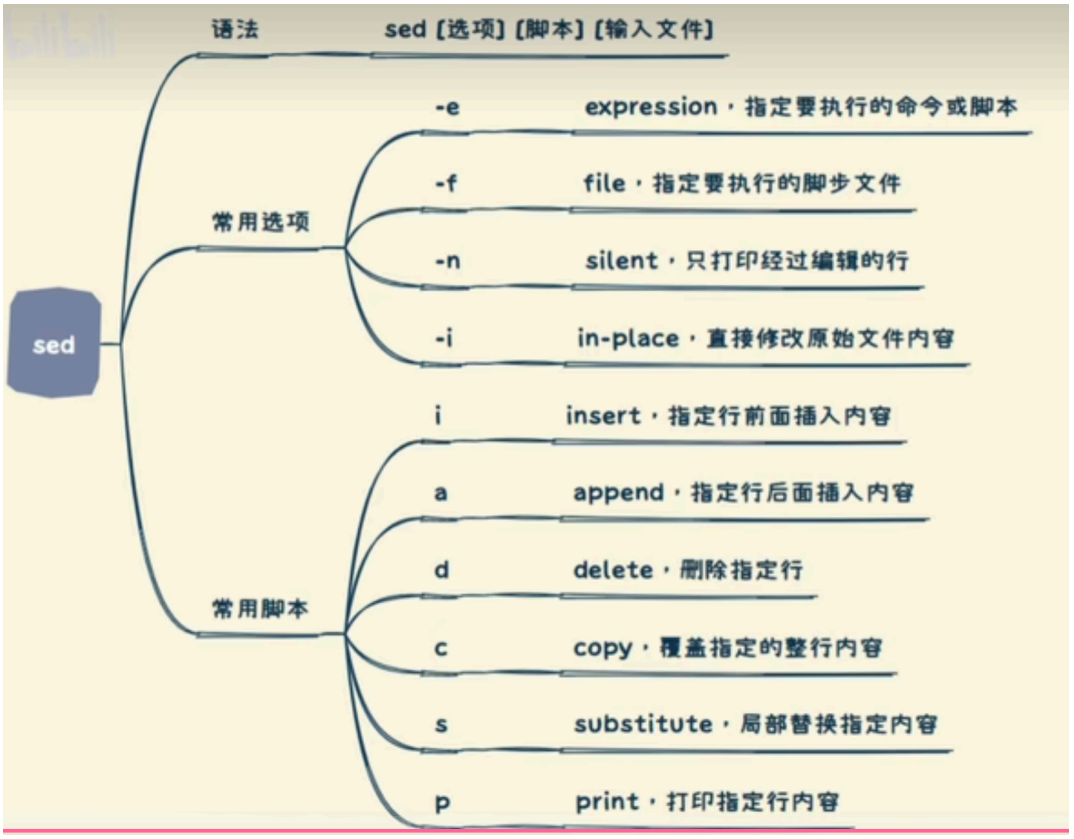
而后者会将 "echo $c>file" 这一字符串输出在终端

二、难点分析

- **Makefile 中变量的应用**

> eg:
>
> ```
> .PHONY: clean all
> CFLAGS = -Wall -g -O2
> targets = hello world
> sources = main.c message.c
> objects = main.o message.o
>
> all: $(targets)
> 	@echo "all done"
>
> $(targets): $(objects)
> 	gcc $(CFLAGS) $(objects) -o $@
>
> %.o: %.c
> 	gcc $(CFLAGS) -c $< -o $@
>
> clean:
> 	rm -f *.o hello world
> ```

- **sed 的用法**


sed
语法
sed [选项] [脚本] [输入文件]
常用选项
-e
expression，指定要执行的命令或脚本
-f
file，指定要执行的脚步文件
-n
silent，只打印经过编辑的行
-i
in-place，直接修改原始文件内容
常用脚本
i
insert，指定行前面插入内容
a
append，指定行后面插入内容
d
delete，删除指定行
c
copy，覆盖指定的整行内容
s
substitute，局部替换指定内容
p
print，打印指定行内容

- **awk的用法**

打印整行：

```
awk '{print}' file
```

打印特定列：

```
awk '{print $1, $2}' file
```

使用分隔符指定列：

```
awk -F',' '{print $1, $2}' file
```

打印行数：

```
awk '{print NR, $0}' file
```

打印行数满足条件的行：

```
awk '/pattern/ {print NR, $0}' file
```

计算列的总和：

```
awk '{sum += $1} END {print sum}' file
```

打印最大值：

```
awk 'max < $1 {max = $1} END {print max}' file
```

格式化输出：

```
awk '{printf "%-10s %-10s\n", $1, $2}' file
```

三、实验体会

> Lab0主要是学习并使用Linux的命令操作，难度不大但是知识点较多。我的体会是熟能生巧，必须要亲自实践才能熟练掌握各种用法。另外要善用搜索引擎，sed、awk是重要的工具，指导书并未过多涉及，需要自己查阅资料学习。
>
> OS是一门重要的课程，我第一次上机的结果并不理想，日后当投入更多的时间精力去学习

四、原创说明

- https://www.runoob.com/linux/linux-comm-sed.html

- https://www.runoob.com/linux/linux-comm-awk.html
- https://www.bilibili.com/video/BV1tyWWeeEpp?t=950.4