

1:

优点

CLI: 节约资源; 程序执行的精度较高; 执行重复性指令方便; 功能强大

GUI: 操作简单; 可呈现各种多媒体数据; 操作易于理解, 不需要去看 man

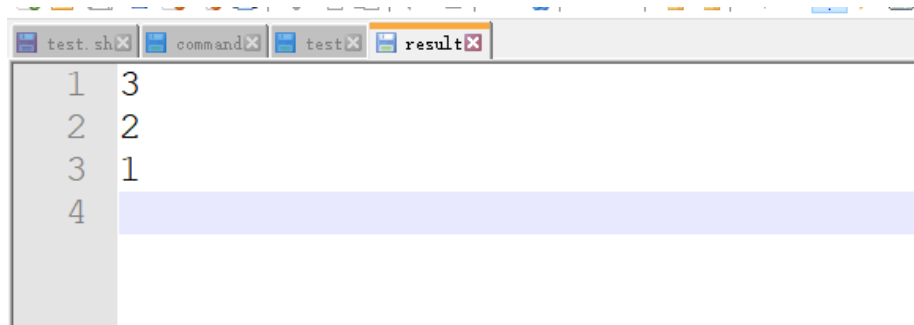
缺点

CLI: 不直观, 对初学者不友好

GUI: 利用的资源较多, 执行重复性指令不方便

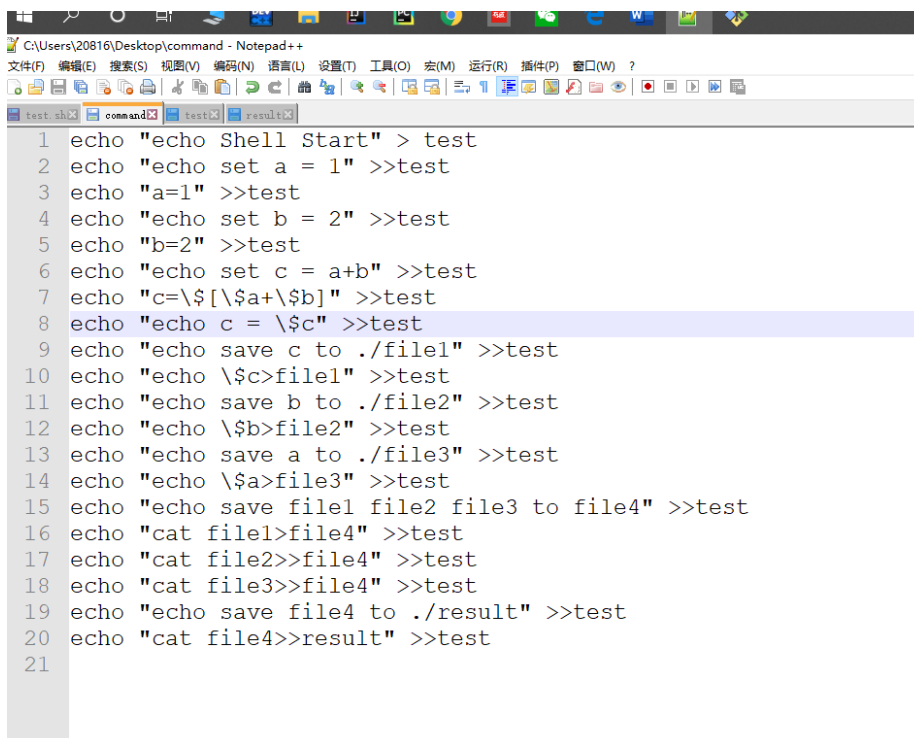
2.

result 中的内容:



```
1 3
2 2
3 1
4
```

command 中的内容:

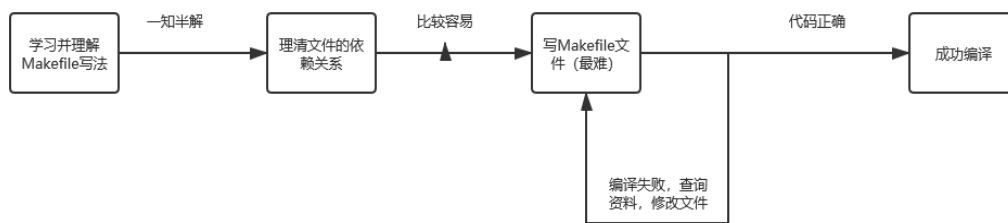


```
1 echo "echo Shell Start" > test
2 echo "echo set a = 1" >>test
3 echo "a=1" >>test
4 echo "echo set b = 2" >>test
5 echo "b=2" >>test
6 echo "echo set c = a+b" >>test
7 echo "c=\${\a+\b}" >>test
8 echo "echo c = \c" >>test
9 echo "echo save c to ./file1" >>test
10 echo "echo \c>file1" >>test
11 echo "echo save b to ./file2" >>test
12 echo "echo \b>file2" >>test
13 echo "echo save a to ./file3" >>test
14 echo "echo \a>file3" >>test
15 echo "echo save file1 file2 file3 to file4" >>test
16 echo "cat file1>file4" >>test
17 echo "cat file2>>file4" >>test
18 echo "cat file3>>file4" >>test
19 echo "echo save file4 to ./result" >>test
20 echo "cat file4>>result" >>test
21
```

echo echo Shell Start 与 echo 'echo Shell Start'效果没有区别

echo echo \c>file1 与 echo 'echo \c>file1' 效果是有区别, echo echo \c>file1 只会将\$c 输入 test 中, 而后者会将单引号中的全部输入 test 中

- 3.
- 分别对应 `git add`
`git add`
`git commit`
- 4.
- `git checkout -- printf.c` 从工作区中删除文件可以从版本库、缓冲区中找回来
`git checkout -- printf.c` 同理
`git rm -- cache Tucao.txt` 删除缓存区的文件
- 5.
- (1) 错误, 克隆时分支并没有被检出, 需要使用 `git checkout <file>` 才能将分支检出, 克隆时只克隆 master 分支
(2) 正确
(3) 正确
(4) 正确
- 6.
- 第一个难点也是最难的点时 Makefile 的编写



第二个难点时 shell 脚本的编写



7. 体会和感想

本次 lab0 作业主要是熟悉 Linux 命令和 git 的操作, git 的操作流程和 shell 的脚本编写较为简单, 差了点资料加上实验指导使得我相关的任务完成得较快, 但是到了任务 8 开始编写多文件的 Makefile 时, 我遇到了较大的困难, 查了大量资料加上和同学们进行讨论, 最后花了较长的时间才将 Makefile 编写正确, 但是还不是完全理解。