课程报告二：Shell 与 Vim

杜培绪

2024 年 9 月 10 日

题目 1.

了解 ls 命令，并加入不同参数使其输出不同的列表。

解决方法：

$$ls\ -a$$

输出所有文件，包含隐藏文件。

$$ls\ -h$$

文件打印以人类可以理解的格式输出，例如使用 454M 而不是 454279954。

$$ls\ -t$$

按照打开的先后顺序输出列表。

$$ls\ --color=auto$$

以彩色文本显示输出结果。

此外，这些参数还可以组合使用，例如

$$ls\ -aht$$

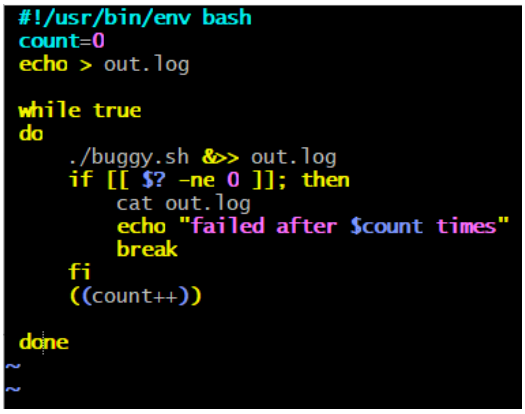
这个命令行能够包含前三种函数的功能。

题目 2.

假设您有一个命令，它很少出错。因此为了在出错时能够对其进行调试，需要花费大量的时间重现错误并捕获输出。编写一段 bash 脚本，运行如下的脚本直到它出错，将它的标准输出和标准错误流记录到文件，并在最后输出所有内容。加分项：报告脚本在失败前共运行了多少次。

解决方法：

创建一个 bash 脚本，在其中编写运行代码。之后运行这个函数，即可

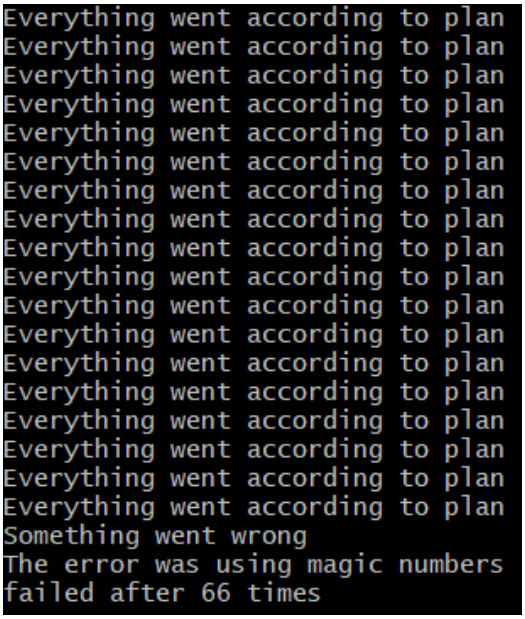


```
#!/usr/bin/env bash
count=0
echo > out.log

while true
do
    ./buggy.sh &>> out.log
    if [[ $? -ne 0 ]]; then
        cat out.log
        echo "failed after $count times"
        break
    fi
    ((count++))

done
~
~
```

得到出错前的执行次数。



```
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Everything went according to plan
Something went wrong
The error was using magic numbers
failed after 66 times
```

题目 3.

编写两个 bash 函数 marco 和 polo 执行下面的操作。每当你执行 marco 时，当前的工作目录应当以某种形式保存，当执行 polo 时，无论现在处在什么目录下，都应当 cd 回到当时执行 marco 的目录。为了方便 debug，你可以把代码写在单独的文件 marco.sh 中，并通过 source marco.sh 命令，(重新）加载函数。

解决方法：

首先创建 marco.sh 文件，并在其中输入所需代码。

您的任务是编写一个命令，它可以递归地查找文件夹中所有的 HTML 文件，并将它们压缩成 zip 文件。注意，即使文件名中包含空格，您的命令也应该能够正确执行 之后重载这个函数。

```
#!/bin/bash
marco(){
    echo "$(pwd)" > $HOME/marco_history.log
    echo "save pwd $(pwd)"
}
polo(){
    cd "$(cat "$HOME/marco_history.log")"
}
```

source marco.sh

之后输入 marco 和 polo 即可运行这两个函数。

题目 4.

编写一个命令，它可以递归地查找文件夹中所有的 HTML 文件，并将它们压缩成 zip 文件。注意，即使文件名中包含空格，您的命令也应该能够正确执行.

解决方法：

首先创建所需文件，创建完毕后返回上一级目录，执行命令

$find\ .\ -type\ f\ -name\ "*.html"\ |\ xargs\ -d\ '\backslash n'\ tar\ -cvzf\ html.zip$

这样就可以查找所有 html 文件并将其压缩为 zip 文件。

```
d@LAPTOP-2BA91PTB MINGW64 ~/learn-git/html_root (main)
$ find . -type f -name "*.html" | xargs -d '\n'  tar -cvzf html.zip
./1.html
./10.html
./2.html
./3.html
./4.html
./5.html
./6.html
./7.html
./8.html
./9.html
./html/xxxx.html
```

题目 5.

请通过在命令行中执行./output x y 输出 x+y 的值，脚本中使用 sum() 函数封装代码并通过调用 sum 函数返回结果。

解决方法：

编写代码如下：

```
#!/bin/bash

sum (){
        echo "第一个参数值为$1"
        echo "第二个参数值为$2"
        sum=$[$1+$2]
        echo $sum
}

sum $1 $2
```

之后按照格式进行输入，能够得到两数之和。

```
$ ./output.sh 10 30
第一个参数值为10
第二个参数值为30
40
```

题目 6.

如何用 shell 查看当前时间?

解决方法:

可以使用 date 命令。

```
d@LAPTOP-2BA91PTB MINGW64 ~/learn-git
$ date
Tue Sep 10 19:58:16     2024
```

date 命令还可以附加参数，date +%Y 输出年份，date +%m 返回月份，date +%F 以 yyyy-mm-dd 格式输出当前时间等。

题目 7.

如何查看电脑储存占用情况?

解决方法:

使用命令 df -h。

```
d@LAPTOP-2BA91PTB MINGW64 ~/learn-git
$ wc -w debug.sh
30 debug.sh
```

题目 8.

shell 中 awk 的用法是什么?

解决方法:

awk 的使用遵循 awk ‘模式或条件 {操作}’ 所需文件的格式。例如用 awk 打印出 word.txt 中的文字，使用命令行

$$awk\ '\{print\}'\ word.txt$$

还可以附加条件，例如

$$awk\ 'NR == 1, NR == 3\{print\}'\ word.txt$$

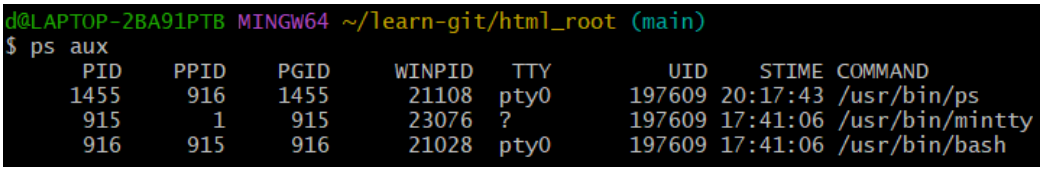
这个命令可以输出文档中前三行的内容。

题目 9.

shell 中的 ps 命令有什么作用?

解决方法:

ps 命令可以用于对系统中进程进行监测控制，它能够显示出进程的状态、内存、CPU 使用情况等。



题目 10.

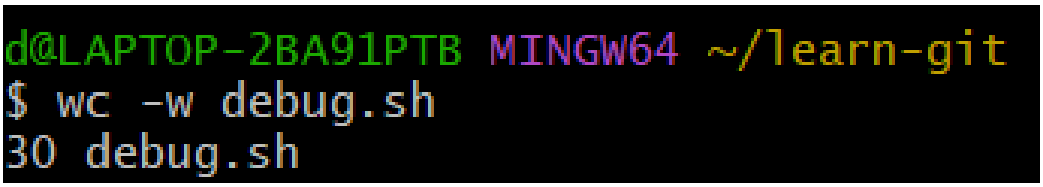
grep 命令有什么作用?

解决方法:

grep 要查找的内容 查找的文件，用这种方式可以在特定文件中查找所需的内容，grep 命令会返回其所在的行。另外 grep 还可以用下述命令在多个文件中进行查找。

$$grep\ "xxxx"\ file_1\ file_2\ file_3$$

grep 可以接收一些参数，如下图所示：

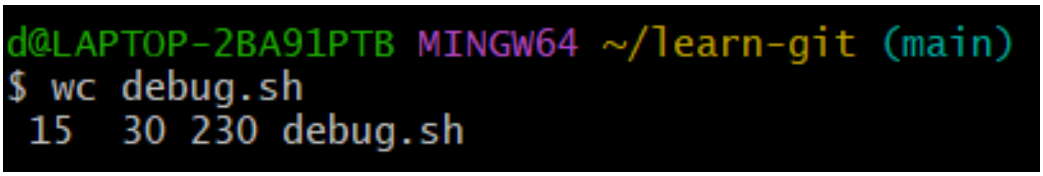


题目 11.

wc 命令的用法

解决方法：

wc 命令可以计算文件的行数、字数、字节数。wc 后面加上-l 表示输出文件的行数，-m 表示字符数，-w 表示字数。如果不加参数则默认全部输出。



题目 12.

cat 命令有什么用处?

解决方法：

cat 的基础用法是查看文件内容。

$$cat\ \ 1.txt$$

即显示 1.txt 的内容。

也可以用 cat 创建文件，与 echo 相似。

$$cat\ \ >\ \ 1.txt$$

用标准输入覆盖原文件内容。如果把 > 改为 »，则意为在原文件末尾输入内容。

还可以用 cat 清除文件内容。

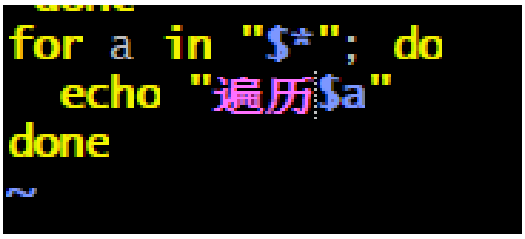
$$cat\ \ /dev/null\ \ >\ \ 1.txt$$

即清空 1.txt 的内容。

题目 13. 如何在 shell 中表示参数?

解决方法:

用 $ 表示参数的数量，$1、$2、$n 表示第几个参数。还可以使用 $*、$@ 表示全部参数，可以用其进行遍历.

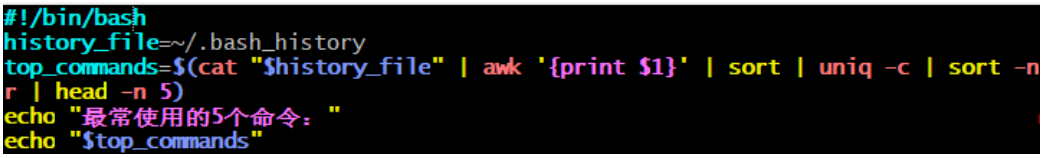


题目 14.

写一个脚本，返回前五条最常用的命令。

解决方法:

使用 awk 命令，编写代码如下：之后尝试运行这个脚本，会发现其成



功输出了最常用的命令，并标注了使用次数。

```
d@LAPTOP-2BA91PTB MINGW64 ~/learn-git (main)
$ ./his.sh
最常使用的5个命令：
    238 git
     42 ls
     36 cd
     26 vim
     18 echo
```

题目 15.

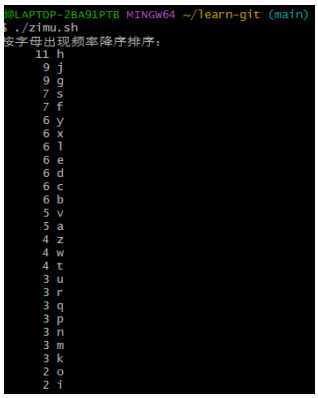
用 shell 脚本处理一段字母，按字母出现次数排序。

解决方法：

可以使用 grep 进行编码，编写代码如下：之后运行代码，脚本将其依

```
#!/bin/bash
content="dwadscfbfgynhjythgfseaszxcfvbgnhjmkilolpoiujhytgrewasxdcfvghyujjklplmlp
qeazxcvhjuytredfghjkljhbgvbhjhgfdsawqertyhgbnmbvcxzxdsxczsqwe"
sort=$(echo "$content" | grep -o . | grep -v '[. ]' | sort | uniq -c | sort -rn)
echo "按字母出现频率降序排序："
echo "$sort"
~
```

次排列出来。



题目 16.

编写一个脚本，计算 300 以内所有能被 7 整除数字的和。

解决方法：

使用循环语句遍历 300 以内所有自然数，判断其能否被 7 整除，代码如下：

```
#!/bin/bash
sum=0
for ((num=1; num<=300; num++))
do
    if [ $((num % 7)) -eq 0 ]; then
        sum=$((sum + num))
    fi
done
echo "和为: $sum"
~
```

题目 17.

写一个脚本，能接受一个参数（文件路径），判断这个参数如果是一个存在的文件就显示“ok”，否则显示“No such file”

解决方法：

使用参数-e 进行判断，编写代码如下：

```
#!/bin/bash
read -p "请输入一个文件路径: " filename
if [ -e $filename ];then
        echo "OK"
else
        echo "No such file"
fi
```

题目 18.

vim 如何切换模式?

解决方法:

用 vim 打开文件后默认为普通模式，按 i、a、o 键进入插入模式，编辑结束后按 ESC 键退出编辑，又回到普通模式。输入：进入末行模式。

题目 19.

用 vim 打开文件时光标默认位于首行，当文件过长或需要准确修改某一行时非常不便，如果解决这个问题?

解决方法:

可以使用命令行

$$vim\ \ name\ \ +n$$

表示打开 name 文件并将光标置于第 n 行。如果去掉行数，改为

$$vim\ \ shadow\ +$$

则表示光标置于末行行首。

题目 20.

末行模式下有哪些命令?

解决方法：

w 表示保存，q 表示退出，wq 为保存并退出，w! 表示强制保存，q! 表示强制退出，wq! 表示强制保存并退出。此外还有 w test.txt，表示将内容保存到 test.txt 中。r test.txt 表示将 test.txt 中的内容复制到当前光标位置等命令。

```
:w 保存文件

:q 退出文件

:wq 保存并退出文件

:w! 强制保存文件

:q!强制退出文件

:wq! 强制保存并退出文件

:w 1.txt 将文件另存到1.txt

:1,3 w 1.txt 将1-3行内容另存到1.txt

:r 2.txt 将2.txt文件内容写入到该文件光标所在行中

:5 r 2.txt 将2.txt文件内容写入到该文件第5行后
```

心得体会：

通过学习 Shell，我了解了 Shell 这个操作系统的作用，并大致了解了 shell 文件下的编程语言。它可以查看与修改各个文件，与 Git 有一定的相似之处。但更有其独特的地方，比如它可以通过编写程序实现更为复杂的功能。

Vim 是一种常用的编辑器。通过对它的学习，我了解了 Vim 中的各种命令，知道了如何在各种模式间进行切换、如何在各种模式下进行操作。

学习这些工具对我未来的学习和工作有着不可忽视的意义，合理利用它们能够提高效率，还能让团队合作变得更为便捷。

Github 链接及提交记录：

GitHub 链接：https://github.com/28935/23020007016

commit 记录截图：

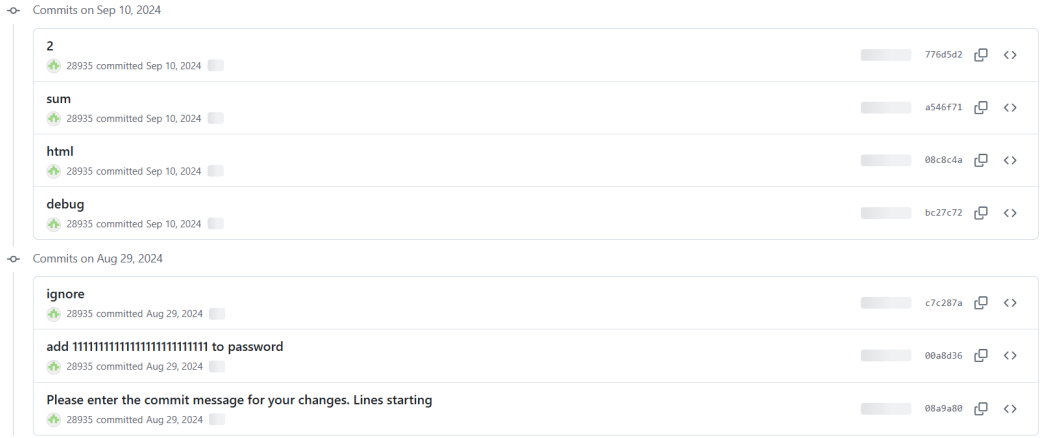


图 1: commit 记录