

shell 工具和脚本和 vim 编辑器实验报告

姓名：刘昱莹 学号：22010022040

2024 年 9 月 2 日

目录

1 练习内容	1
1.1 shell 工具和脚本	1
1.2 vim 编辑器	1
2 结果	1
2.1 shell 工具和脚本	1
2.1.1 习题 1	1
2.1.2 习题 2	2
2.1.3 习题 3	3
2.1.4 习题 4	4
2.1.5 习题 5	4
2.2 vim 编辑器	5
3 解题感悟	5
4 Github 链接	5

1 练习内容

1.1 shell 工具和脚本

5 个课后习题。

1.2 vim 编辑器

练习使用 vim 编辑器中的各种命令并用其编辑 shell 脚本。

2 结果

2.1 shell 工具和脚本

2.1.1 习题 1

ls 命令的用法：

- 所有文件（包括隐藏文件）：-a
- 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）：-h
- 文件以最近访问顺序排序：-t
- 以彩色文本显示输出结果：-color

```
lly@lly-virtual-machine:~/Desktop$ ls -l -a -t -h --color=auto
total 48K
drwxr-x--- 15 lyy lyy 4.0K  9月  2 15:20 ..
drwxrwxr-x  2 lyy lyy 4.0K  8月 30 11:28 bar
drwxrwxr-x  2 lyy lyy 4.0K  8月 30 11:27 foo
drwxr-xr-x  6 lyy lyy 4.0K  8月 30 11:22 .
-rwxrwxr-x  1 lyy lyy 356  8月 30 11:10 test.sh
-rw-rw-r--  1 lyy lyy 283  8月 30 10:55 marco.sh
drwxrwxr-x  3 lyy lyy 4.0K  8月 30 09:09 Learn-Git-and-LaTeX
drwxrwxr-x  3 lyy lyy 4.0K  8月 23 11:31 learnGit
-rw-rw-r--  1 lyy lyy 1.7K  6月 22 15:57 rsa_code.py
-rw-rw-r--  1 lyy lyy 758  6月 22 15:29 chacha_code.py
-rw-rw-r--  1 lyy lyy 2.3K  6月 22 15:25 code.py
-rw-rw-r--  1 lyy lyy 768  6月 22 14:46 code1.py
```

图 1: ls 命令输出结果

2.1.2 习题 2

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。

Listing 1: marco.sh

```
1 #!/bin/bash
2
3 marco() {
4     export MARCO_DIR=$(pwd)
5     echo $MARCO_DIR
6 }
7
8 polo() {
9     if [ -z "$MARCO_DIR" ]; then
10        echo "No directory has been saved. Please run marco first."
11    else
12        cd "$MARCO_DIR" || echo "Failed to change directory to $MARCO_DIR"
13    fi
14 }
```

```
lly@lly-virtual-machine:~/Desktop$ source marco.sh
lly@lly-virtual-machine:~/Desktop$ marco
/home/lyy/Desktop
lly@lly-virtual-machine:~/Desktop$ cd
lly@lly-virtual-machine:~$ polo
lly@lly-virtual-machine:~/Desktop$
```

图 2: marco.sh 运行结果

2.1.3 习题 3

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。

Listing 2: buggy.sh

```
1 #!/usr/bin/env bash
2
3 n=$(( RANDOM % 100 ))
4
5 if [[ n -eq 42 ]]; then
6     echo "Something went wrong"
7     >&2 echo "The error was using magic numbers"
8     exit 1
9 fi
10
11 echo "Everything went according to plan"
```

Listing 3: debug.sh

```
1 #!/usr/bin/env bash
2 count=0
3 echo > out.log
4
5 while true
6 do
7     ./buggy.sh &>> out.log # 将buggy.sh的标准输出和标准错误追加到到out.log文件中
8     if [[ $? -ne 0 ]]; then # buggy.sh文件执行错误
9         cat out.log
10        echo "failed after $count times"
11        break
12    fi
13    ((count++))
14
15 done
```

```
lly@lly-virtual-machine:~/Desktop$ vim debug.sh
lly@lly-virtual-machine:~/Desktop$ vim buggy.sh
lly@lly-virtual-machine:~/Desktop$ ./debug.sh

./debug.sh: line 7: ./buggy.sh: Permission denied
failed after 0 times
lly@lly-virtual-machine:~/Desktop$ chmod +x buggy.sh
lly@lly-virtual-machine:~/Desktop$ ./debug.sh

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 17 times
lly@lly-virtual-machine:~/Desktop$
```

图 3: debug.sh 运行结果

2.1.4 习题 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。

```
lly@lly-virtual-machine: ~/Desktop/html_root$ ls
1.html 2.html 3.html 4.html 5.html 6.html 7.html 8.html 9.html html
lly@lly-virtual-machine:~/Desktop/html_root$ ^C
lly@lly-virtual-machine:~/Desktop/html_root$ find . -type f -name "*.html" | xargs -d '\n' tar -cvzf html.zip
./2.html
./1.html
./8.html
./3.html
./7.html
./9.html
./html/xxxx.html
./6.html
./4.html
./5.html
lly@lly-virtual-machine:~/Desktop/html_root$ ls
1.html 2.html 3.html 4.html 5.html 6.html 7.html 8.html 9.html html html.zip
```

2.1.5 习题 5

编写一个命令或脚本递归的查找文件夹中最近使用的文件。更通用的做法，你可以按照最近的使用时间列出文件吗？

```

lyy@lyy-virtual-machine:~/Desktop$ find . -type f -mmin -60 -print0 | xargs -0 ls -lt | head -1
-rw-rw-r-- 1 lyy lyy 197  9月  2 19:40 ./html_root/html.zip
lyy@lyy-virtual-machine:~/Desktop$ find . -type f -mmin -60 -print0 | xargs -0 ls -lt
-rw-rw-r-- 1 lyy lyy 197  9月  2 19:40 ./html_root/html.zip
-rw-rw-r-- 1 lyy lyy 115  9月  2 19:33 ./html_root/html/html.zip
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:30 ./html_root/html/xxxx.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/9.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/1.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/2.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/3.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/4.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/5.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/6.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/7.html
-rw-rw-r-- 1 lyy lyy  0  9月  2 19:29 ./html_root/8.html

```

2.2 vim 编辑器

```

#!/bin/bash
echo "Hello Home!"
for file in $(ls ~/Desktop)
do
    echo $file
done

myUrl1="https://www.google.com"
readonly myUrl1
myUrl1="https://www.baidu.com"
echo $myUrl1

myname="yuying"
my_string_1="Hello, ${myname}!\n"
my_string_2='Hello, ${myname}!'
echo -e ${my_string_1}$my_string_2
echo ${#my_string_1} ${#my_string_2}
find . -name '*.py' -type f
~
~

```

3 解题感悟

通过本次实验，我学习到了一些 shell 工具的用法，例如：man、grep、find、xargs 以及管道等。我还简单学习了 bash 脚本。我也了解了更多关于 vim 编辑器的知识，之前只是简单知道命令行模式和插入模式、退出和保存并退出，通过这次学习，我学到了更多 vim 中的指令，我之后使用 vim 编辑器会更加顺畅。

4 Github 链接

<https://github.com/yuying019828/shell-vim>