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 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

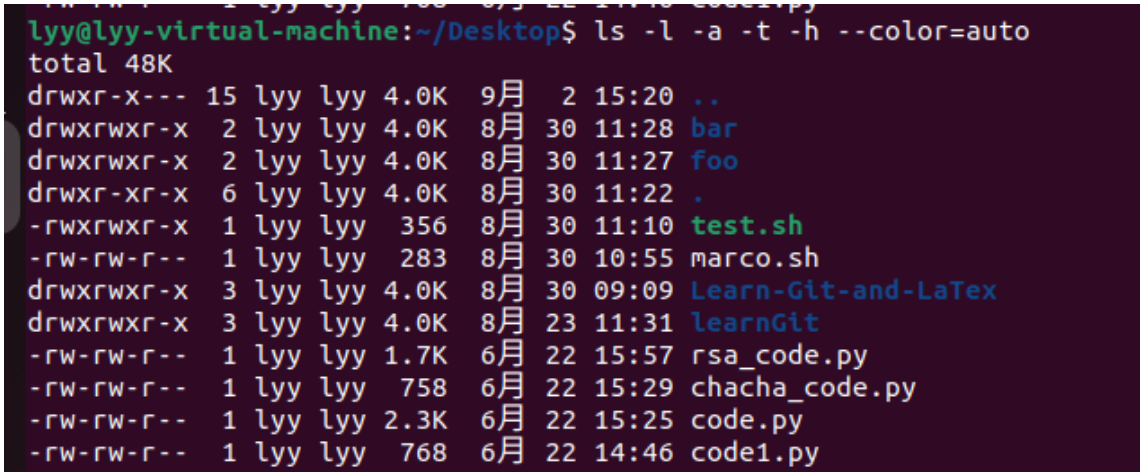


图 1: ls 命令输出结果

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

```
#!/bin/bash

marco() {
    export MARCO_DIR=$(pwd)
    echo $MARCO_DIR
}

polo() {
    if [ -z "$MARCO_DIR" ]; then
        echo "No directory has been saved. Please run marco first."
    else
        cd "$MARCO_DIR" || echo "Failed to change directory to $MARCO_DIR"
    fi
}
```

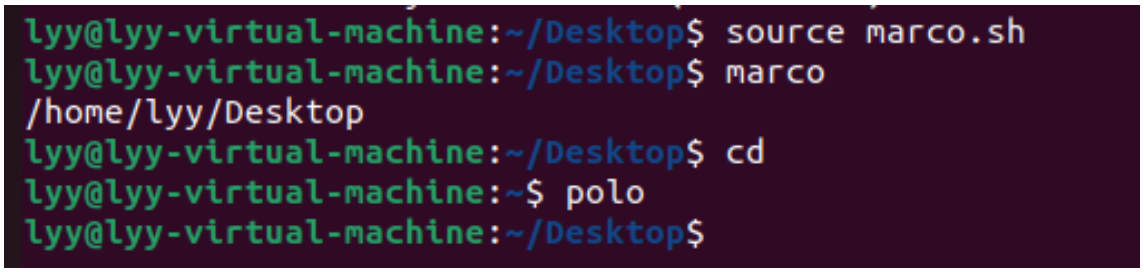


图 2: marco.sh 运行结果

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

```
#!/usr/bin/env bash

n=$(( RANDOM % 100 ))

if [[ n -eq 42 ]]; then
    echo "Something went wrong"
    >&2 echo "The error was using magic numbers"
    exit 1
fi

echo "Everything went according to plan"
```

Listing 3: debug.sh

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

while true
do
    ./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
    if [[ $? -ne 0 ]]; then  # buggy.sh文件执行错误
        cat out.log
        echo "failed after $count times"
        break
    fi
    ((count++))

done
```

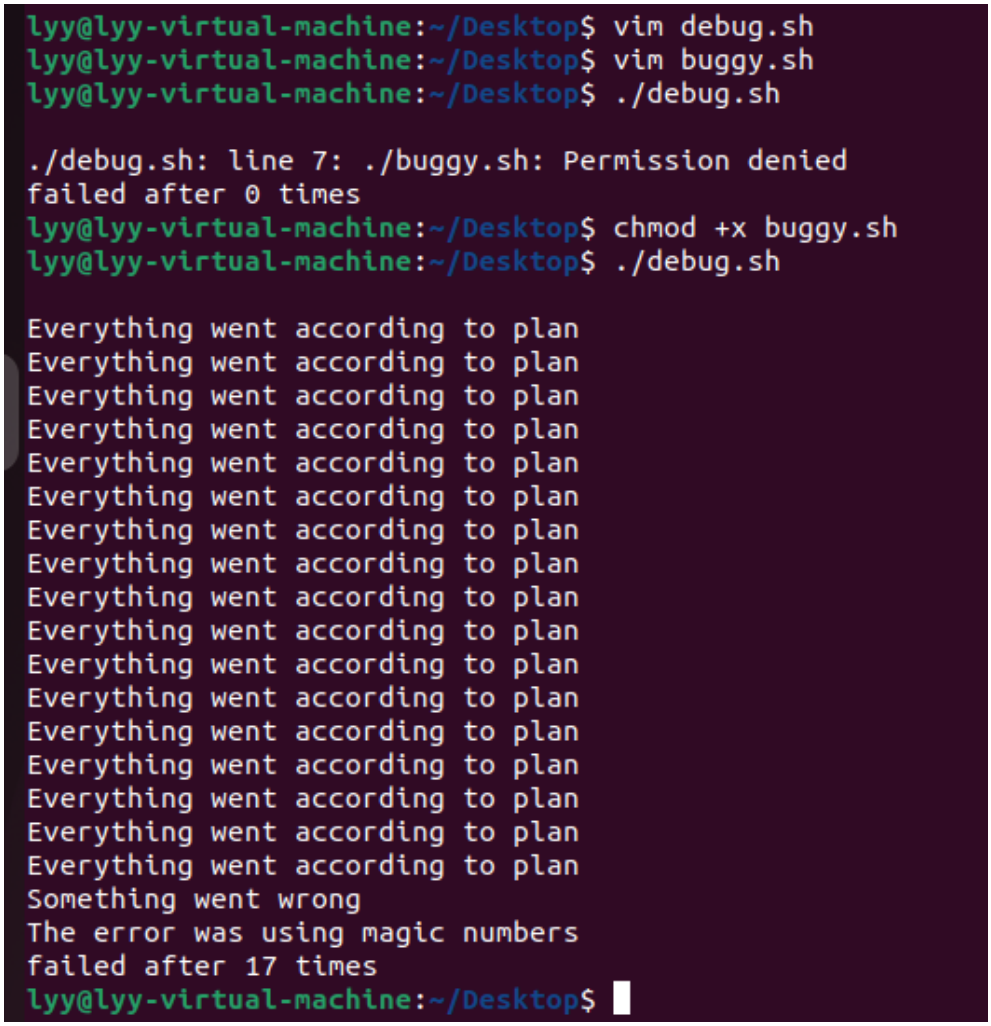
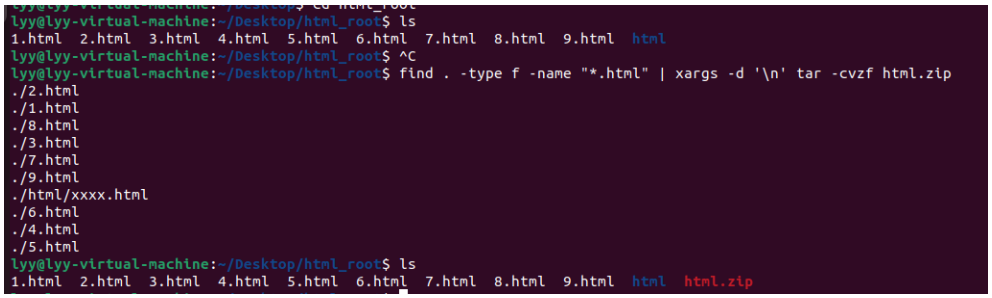


图 3: debug.sh 运行结果

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。



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。我也了解到了更多关于 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`