

# 实验报告 2

姓名：李燮珍

学号：22020007048

实验时间：2024/08/30

## 一. 实验板块

Shell 工具和脚本

编辑器 (Vim)

数据整理

## 二. 实验内容

### 1.Shell 工具和脚本相关：

1. 阅读 man ls , 然后使用 ls 命令进行如下操作：

- 所有文件（包括隐藏文件）
- 文件打印以人类可以理解的格式输出
- 文件以最近访问顺序排序
- 以彩色文本显示输出结果

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。

3. 了解 shell 脚本

4. 编写一段 bash 脚本，运行相关脚本直到它出错，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。

5. 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。

6. 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。

7. 练习 grep 命令来查找代码
8. 查找 shell 命令

## 2. 编辑器 Vim 相关:

1. 了解 vim 编辑模式
2. 练习 vim 的基本操作
3. 自定义 Vim
4. 扩展 Vim

## 3. 数据整理相关:

1. 正则表达式
2. awk - 另外一种编辑器
3. 分析数据
4. 利用数据整理来确定参数
5. 整理二进制数据

# 三. 实验过程

Shell 工具和脚本相关实验:

## 1. 显示所有文件

代码: `ls -a`

## 2. 文件打印以人类可以理解的方式

代码: `ls -h`

## 3. 文件以最近访问顺序排序

代码: `ls -t`

## 4. 文件以彩色文本显示输出结构

代码: `ls -color=auto`

以上四个实验截图;

```
lzx@lzx-virtual-machine:~$ man ls
lzx@lzx-virtual-machine:~$ ls -a
. .config .ICEauthority .Xauthority
.. .dbus .local .xsession-errors
.bash_logout examples.desktop .mozilla .xsession-errors
.bashrc .gconf .presage 公共的
.cache .gnupg .profile 模板
LibreOfficeWriter machine:~$ ls -h
examples.desktop 公共的 模板 视频 图片 文档 下载 音乐
lzx@lzx-virtual-machine:~$ ls -t
公共的 模板 视频 图片 文档 下载 音乐 桌面 examples.de + 0.0
lzx@lzx-virtual-machine:~$ ls --color=auto
examples.desktop 公共的 模板 视频 图片 文档 下载 音乐
```

## 5. 典型输出

代码: `ls -l -a`

```
lzx@lzx-virtual-machine:~$ ls -l -a
总用量 112
drwxr-xr-x 18 lzx lzx 4096 8月 30 08:39 .
drwxr-xr-x 3 root root 4096 12月 22 2023 ..
-rw-r--r-- 1 lzx lzx 220 12月 22 2023 .bash_logout
-rw-r--r-- 1 lzx lzx 3771 12月 22 2023 .bashrc
drwx----- 13 lzx lzx 4096 8月 30 08:39 .cache
drwx----- 17 lzx lzx 4096 12月 22 2023 .config
drwx----- 3 lzx lzx 4096 12月 22 2023 .dbus
-rw-r--r-- 1 lzx lzx 8980 12月 22 2023 examples.desktop
drwx----- 2 lzx lzx 4096 8月 30 08:39 .gconf
drwx----- 3 lzx lzx 4096 8月 30 08:38 .gnupg
-rw----- 1 lzx lzx 1110 8月 30 08:38 .ICEauthority
-rw----- 1 lzx lzx 4096 12月 22 2023 .local
LibreOffice Calc lzx lzx 4096 8月 30 08:39 .mozilla
drwx----- 2 lzx lzx 4096 12月 22 2023 .presage
-rw-r--r-- 1 lzx lzx 655 12月 22 2023 .profile
-rw----- 1 lzx lzx 64 8月 30 08:38 .xauthority
-rw----- 1 lzx lzx 84 8月 30 08:38 .xsession-errors
-rw----- 1 lzx lzx 84 8月 1 10:48 .xsession-errors.ol
drwxr-xr-x 2 lzx lzx 4096 12月 22 2023 公共的
drwxr-xr-x 2 lzx lzx 4096 12月 22 2023 模板
drwxr-xr-x 2 lzx lzx 4096 12月 22 2023 视频
drwxr-xr-x 2 lzx lzx 4096 12月 22 2023 图片
```

## 6. 编辑 marco.sh 文件

代码: `vim marco.sh`

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

## 7. 加载函数并执行 polo

代码:

`source marco.sh`

`marco`

cd

```
lxz@lxz-virtual-machine:~/文档$ vim marco.sh
lxz@lxz-virtual-machine:~/文档$ ls
marco.sh
lxz@lxz-virtual-machine:~/文档$ source marco.sh
lxz@lxz-virtual-machine:~/文档$ marco
lxz@lxz-virtual-machine:~/文档$ cd
lxz@lxz-virtual-machine:~$ polo
lxz@lxz-virtual-machine:~/文档$
```

## 7. 编写 shell 脚本

buggy.sh 脚本

```
n=$((RANDOM % 100))
if [[ n -eq 42 ]]; then
    echo "Something went wrong"
    >&2 echo "The error was using magic numbers"
    exit 1
fi
echo "Everthing went according to plan"
```

while 循环

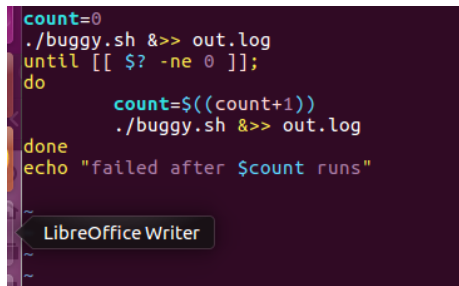
```
count=0
echo > out.log
while true
do
    ./buggy.sh &>> out.log
    if [[ $? -ne 0 ]];then
        cat out.log
        echo "failed after $count times"
        break
    fi
    ((count++))
done
```

for 循环

```
for((count=0;;count++))
do
    ./buggy.sh &>> out.log
    if [[ $? -ne 0 ]];then
        echo "failed after $count times"
        break
    fi
done
./buggy.sh &>> out.log
until [[ $? -ne 0 ]];
do
    count=$((count+1))
    ./buggy.sh &>> out.log
done
```

until 脚本

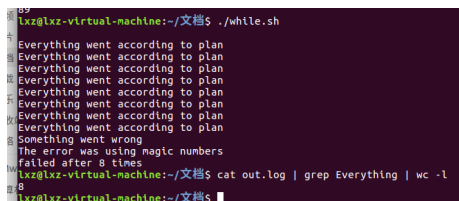
```
count=0
./buggy.sh &>> out.log
until [[ $? -ne 0 ]];
do
    count=$((count+1))
    ./buggy.sh &>> out.log
done
echo "failed after $count runs"
```



## 8. 执行测试脚本, 并验证脚本结果的正确性

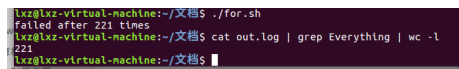
while 脚本测试结构:

```
lxz@lxz-virtual-machine:~/文档$ ./while.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
Something went wrong
The error was using magic numbers
failed after 8 times
lxz@lxz-virtual-machine:~/文档$ cat out.log | grep Everything | wc -l
8
lxz@lxz-virtual-machine:~/文档$
```



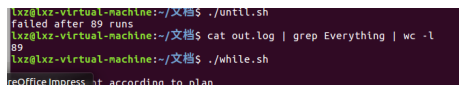
for 脚本测试结构:

```
lxz@lxz-virtual-machine:~/文档$ ./for.sh
failed after 221 times
lxz@lxz-virtual-machine:~/文档$ cat out.log | grep Everything | wc -l
221
lxz@lxz-virtual-machine:~/文档$
```



until 脚本测试结构:

```
lxz@lxz-virtual-machine:~/文档$ ./until.sh
failed after 89 runs
lxz@lxz-virtual-machine:~/文档$ cat out.log | grep Everything | wc -l
89
lxz@lxz-virtual-machine:~/文档$ ./while.sh
Everything went according to plan
```



## 9. 创建所需要 html 文件

代码:

```
mkdir html_root
```

```
cd html_root
```

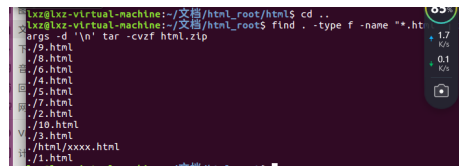
```
touch {1..10}.html
```

```
mkdir html
```

```
cd html
touch xxxx.html
```

## 9. 执行 find 命令

执行结果：



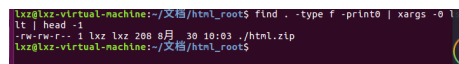
```
lxz@lxz-virtual-machine:~/文档/html_root/html$ cd ..
lxz@lxz-virtual-machine:~/文档/html_root$ find . -type f -name '*.html' | xargs -d '\n' tar -cvzf html.zip
./9.html
./8.html
./6.html
./4.html
./5.html
./7.html
./2.html
./10.html
./3.html
./html/xxxx.html
./1.html
```

## 10. 查找文件夹中最近使用的文件

代码：

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

执行结果：



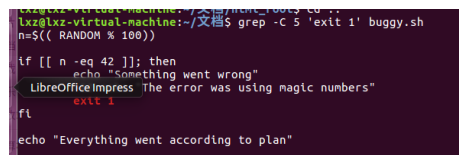
```
lxz@lxz-virtual-machine:~/文档/html_root$ find . -type f -print0 | xargs -0 | head -1
./html/xxxx.html
lxz@lxz-virtual-machine:~/文档/html_root$
```

## 11 使用 grep -C 获取查找结果的上下文 (Context)

代码：

```
grep -C 5 'exit 1' buggy.sh
```

例子（在 buggy.sh 文件中找出匹配 exit 1 的前后五行）：



```
lxz@lxz-virtual-machine:~/文档/html_root$ cd ..
lxz@lxz-virtual-machine:~/文档$ grep -C 5 'exit 1' buggy.sh
n=$(( RANDOM % 100))

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

echo "Everything went according to plan"
```

编辑器 Vim 相关实验：

## 12. 完成 vimtutor

代码：

```
vimtutor
```

```
= Welcome to the VIM Tutor - Version 1.7 =
-----
Vim is a very powerful editor that has many commands, too many to
explain in a tutor such as this. This tutor is designed to describe
enough of the commands that you will be able to easily use Vim as
an all-purpose editor.

The approximate time required to complete the tutor is 30 minutes,
depending upon how much time is spent with experimentation.

ATTENTION:
The commands in the lessons will modify the text. Make a copy of this
file to practice on (if you started "vintutor" this is already a copy).

It is important to remember that this tutor is set up to teach by
use. That means that you need to execute the commands to learn them
properly. If you only read the text, you will forget the commands!

Now, make sure that your Caps-Lock key is NOT depressed and press
the j key enough times to move the cursor so that lesson 1.1
completely fills the screen.
-----
Lesson 1.1: MOVING THE CURSOR

** To move the cursor, press the h,j,k,l keys as indicated. **
      A
      k             Hint: The h key is at the left and moves left.
    < h   l >       The l key is at the right and moves right.
      j             The j key looks like a down arrow.
      v
1. Move the cursor around the screen until you are comfortable.
2. Hold down the down key (j) until it repeats.
   Now you know how to move to the next lesson.
3. Using the down key, move to lesson 1.2.

NOTE: If you are ever unsure about something you typed, press <ESC> to place
you in Normal mode. Then retype the command you wanted.

NOTE: The cursor keys should also work. But using hjkl you will be able to
move around much faster, once you get used to it. Really!
-----
Lesson 1.2: EXITING VIM

!! NOTE: Before executing any of the steps below, read this entire lesson!!
1. Press the <ESC> key (to make sure you are in Normal mode).
2. Type:      :q! <ENTER>.
   This exits the editor, DISCARDING any changes you have made.
```

### 13. 下载 vimrc，然后把它保存到 ~/.vimrc

代码：

mv vimrc ~/.vimrc

```
kzm@DESKTOP-CCFC497 MINGW64 ~/git/missing-semester-cn.github.io/副本/missing-s
missing-semester-cn.github.io (master)
$ ls
404.html      _2020/          favicon-16x16.png  robots.txt
CNAME         _config.yml     favicon-32x32.png  static/
Gemfile       _includes/      favicon.ico        vimrc
Gemfile.lock  _layouts/       index.md           第一次实验报告.pdf
README.md     about.md        lectures.html
2019/         apple-touch-icon.png  license.md
```

```
kzm@DESKTOP-CCFC497 MINGW64 ~/git/missing-semester-cn.github.io/副本/missing-s
missing-semester-cn.github.io (master)
$ mv vimrc ~/.vimrc
```

### 14. 创建插件文件夹

代码：mkdir -p ~/.vim/pack/vendor/start

### 15. 下载这个插件

代码：cd ~/.vim/pack/vendor/start

git clone https://github.com/ctrlpvim/ctrlp.vim

以上两个操作如图：

```

missing-semester-cn.github.io (master)
$ mv vimrc ~/.vimrc
lxzm@DESKTOP-CCFC49T MINGW64 ~/git/missing-semester-cn.github.io/副本/missing-semester-cn.github.io (master)
$ mkdir -p ~/.vim/pack/vendor/start
lxzm@DESKTOP-CCFC49T MINGW64 ~/git/missing-semester-cn.github.io/副本/missing-semester-cn.github.io (master)
$ cd ~/.vim/pack/vendor/start
lxzm@DESKTOP-CCFC49T MINGW64 ~/.vim/pack/vendor/start
$ git clone https://github.com/ctrlpvim/ctrlp.vim
Cloning into 'ctrlp.vim'...
remote: Enumerating objects: 4299, done.
remote: Counting objects: 100% (168/168), done.
remote: Compressing objects: 100% (105/105), done.
remote: Total 4299 (delta 71), reused 147 (delta 62), pack-reused 4131 (from 1)
Receiving objects: 100% (4299/4299), 1.70 MiB | 349.00 KiB/s, done.
Resolving deltas: 100% (1661/1661), done.
lxzm@DESKTOP-CCFC49T MINGW64 ~/.vim/pack/vendor/start

```

## 16. 在 ~/.vimrc 中添加设置

代码:

```
set runtimepath^=~/.vim/pack/vendor/start/ctrlp.vim
```

```

12 " bad habit. The former is enforceable through a .vimrc, while
11 " how to prevent the latter.
10 " Do this in normal mode.
9 noremap <Left> :echo "Use h"<CR>
8 noremap <Right> :echo "Use l"<CR>
7 noremap <Up> :echo "Use k"<CR>
6 noremap <Down> :echo "Use j"<CR>
5 " ...and in insert mode
4 inoremap <Left> <ESC>:echo "Use h"<CR>
3 inoremap <Right> <ESC>:echo "Use l"<CR>
2 inoremap <Up> <ESC>:echo "Use k"<CR>
1 set runtimepath A=~/.vim/pack/vendor/start/ctrlp.vim
81

```

## 17. 用 CtrlP 来在一个工程文件夹里定位一个文件

```

[No Name] [unix] (07:59 01/01/1970)
> ctrlp.vim/autoload/ctrlp.vim
> ctrlp.vim/plugin/ctrlp.vim
> ctrlp.vim/doc/ctrlp.txt
> ctrlp.vim/doc/ctrlp.cnx
> ctrlp.vim/readme.md
> ctrlp.vim/autoload/ctrlp/undo.vim
> ctrlp.vim/autoload/ctrlp/line.vim
> ctrlp.vim/autoload/ctrlp/tag.vim
> ctrlp.vim/autoload/ctrlp/dir.vim
> ctrlp.vim/LICENSE
prt_path {<br>=<buf> <-> /c:/Users/lxzm/.vim
>>> vim_

```

## 18. 自定义 CtrlP

添加以下代码到 ~/.vimrc 来用按 Ctrl-P 打开 CtrlP

```
let g:ctrlp_map = '<c-p>'
```

```
let g:ctrlp_cmd = 'CtrlP'
```

```
let g:ctrlp_working_path_mode = 'ra'
```

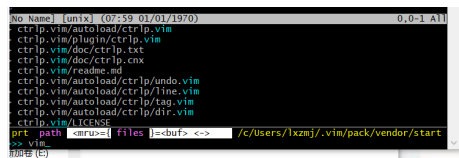
```

4 inoremap <up> <ESC>:echo "Use k"<CR>
3 set runtimepath A=~/.vim/pack/vendor/start/ctrlp.vim
2 let g:ctrlp_map = '<c-p>'
1 let g:ctrlp_cmd = 'CtrlP'
3 let g:ctrlp_working_path_mode = 'ra'
2 vimrc[1] [unix] [11-21 20/08/2024]

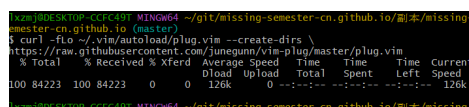
```



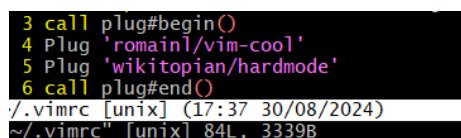
此时打开 vim 直接按键 Ctrl -P 的结果如同



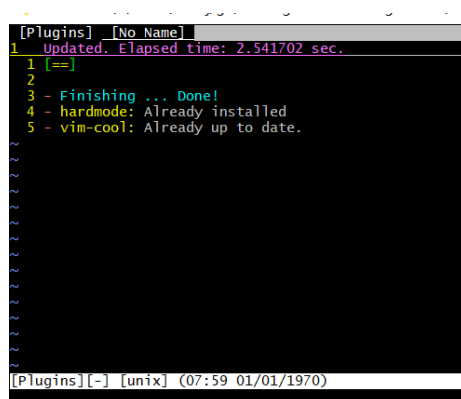
## 19. 安装 vim-plug



## 20. 修改 `~/.vimrc`，在文件中插入要安装的插件



## 21. 在 vim 命令行中执行:PlugInstall 来安装插件



### 数据整理相关实验:

## 22. 统计 words 文件中包含至少三个 a 且不以's 结尾的单词个数。

编写 words 文件:



执行查询语句代码:

```
cat words | tr "[:upper:]" "[:lower:]" | grep -E "^(([a]*a){3}.*$" | grep -v "'s$" | wc -l
```

```
lxz@lxz-virtual-machine:~/文档$ cat words | tr "[:upper:]" "[:lower:]" | grep -E "^(([a]*a){3}.*$" | grep -v "'s$" | wc -l
2
lxz@lxz-virtual-machine:~/文档$
```

## 四. 实验中遇到的问题以及解决办法

### 1. 在实验 8 执行文件时权限不够

在执行测试脚本时,发现各个测试脚本的文件都没有权限,后上网学习了相关 chmod 的用法,更改了几个文件的权限后成功执行。

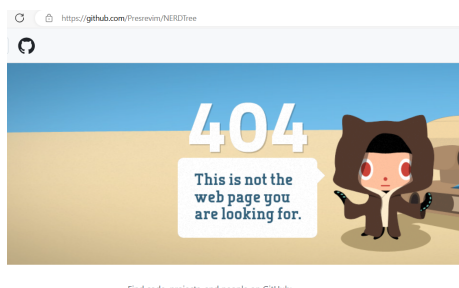
```
lxz@lxz-virtual-machine:~/文档$ ./for.sh
bash: ./for.sh: 权限不够
lxz@lxz-virtual-machine:~/文档$
lxz@lxz-virtual-machine:~/文档$ chmod +x for.sh
lxz@lxz-virtual-machine:~/文档$ ./for.sh
failed after 295 times
lxz@lxz-virtual-machine:~/文档$
```

### 2. 实验 21 在用 PlugInstall 安装插件时失败

vim 命令行中执行:PlugInstall 时安装插件失败,显示找不到所需安装插件的网站。

```
[Plugins] [No Name]
8 updated. Elapsed time: 3.013203 sec.
7 [-x]
6
5 - Finishing ... Done!
4 - hardmode: Already up to date.
3 KNEROTree
2 Cloning into 'C:/Users/lxzmj/.vim/plugged/NEROTree'...
1 remote: Repository not found.
fatal: repository 'https://github.com/Presrevim/NEROTree.git/' not found
```

后在 github 网站搜寻果然没有了这个插件的网站



于是再次修改修改 ~/.vimrc, 改变了在文件中要安装的插件

```
3 call plug#begin()
4 Plug 'romainl/vim-cool'
5 Plug 'wikitopian/hardmode'
6 call plug#end()
~/.vimrc [unix] (17:37 30/08/2024)
~/.vimrc" [unix] 84L, 3339B
```

最后成功下载插件

## 五. 实验收获与心得

本次的实验学习了 Shell 工具和脚本, 编辑器 (Vim), 数据整理三个大模块。其中特别详细的学习了 shell 和 Vim 的相关用法。特别是 Shell 是在虚拟机上操作的, 不仅让我回忆了虚拟机的相关知识, 也让我对 shell 的部分命令有所掌握, 如查找文件和查找代码的相关命令, 还有 shell 脚本也是一个较为新的知识点, 通过这一次实验中编辑的脚本, 让我对 shell 脚本语言也有所了解。Vim 编辑器算是我们实验中经常用到的一个编辑器, 通过这一次实验不仅让我复习了 vim 的相关基本知识, 也让我了解到了它一些特殊功能, 如自定义 CtrlP。总之这一次的实验知识很丰富, 让我也收获很多, 我相信这对我之后的实验都是有所帮助的。仓库链接: <https://github.com/lixiezhn/System-Development-Tool->