

实验报告

彭沛粟

September 12, 2025

Contents

1	练习内容	1
1.1	<i>Shell</i> 工具和脚本	1
1.2	<i>vim</i> 编辑器	7
1.3	数据整理	10
2	解题感悟	12
2.1	<i>Shell</i> 工具和脚本	12
2.2	<i>vim</i> 编辑器	12
2.3	数据整理	12

1

练习内容

1.1 **Shell** 工具和脚本

1、实例 01：在 /tmp 下新建一个名为 **missing** 的文件夹。

- (1) 打开/tmp 文件夹发现并不存在，所以先在主目录添加一个名为 tmp 的文件夹
- (2) 然后输入 cd tmp 进入对应目录
- (3) 输入 mkdir missing 创建一个新的名为 missing 的文件夹
- (4) 输入 ls，检查该目录下 missing 文件夹是否已存在1

```
peng@peng-virtual-machine:~$ cd tmp  
bash: cd: tmp: No such file or directory  
peng@peng-virtual-machine:~$ cd tmp  
peng@peng-virtual-machine:~/tmp$ mkdir missing  
peng@peng-virtual-machine:~/tmp$ ls  
missing  
peng@peng-virtual-machine:~/tmp$
```

Figure 1: 如图所示

2、实例 02：用 **man** 查看程序 **touch** 的使用手册

(1) 输入命令：man touch²

```

TOUCH(1)                               User Commands                               TOUCH(1)

NAME
    touch - change file timestamps

SYNOPSIS
    touch [OPTION]... FILE...

DESCRIPTION
    Update the access and modification times of each FILE to the current
    time.

    A FILE argument that does not exist is created empty, unless -c or -h
    is supplied.

    A FILE argument string of - is handled specially and causes touch to
    change the times of the file associated with standard output.

    Mandatory arguments to long options are mandatory for short options
    too.

    -a      change only the access time

Manual page touch(1) line 1 (press h for help or q to quit)

```

Figure 2: 如图所示

3、实例 03：用 **touch** 在 **missing** 文件夹中新建一个叫 **semester** 的文件。并将图中内容一行一行地写入 **semester** 文件

- (1) 打开 **missing** 文件夹的目录，新建文件^{??}: **semester**¹
- (2) 使用 **echo** 向 **semester** 输入 “#!/bin/sh”²
- (3) 使用 » 向文件追加内容 “curl –head –silent https://missing.csail.mit.edu”
- (4) 使用 **cat** 检查一下内容是否已经成功写入 **semester** 中⁴

```

peng-peisu@pengpeisu:~$ cd tmp/missing
peng-peisu@pengpeisu:~/tmp/missing$ touch semester
peng-peisu@pengpeisu:~/tmp/missing$ ls
semester

```

¹ cd tmp/missing

² 注意！# 的意义是注释，需要有引号包裹，而! 在双引号包裹下仍具有特殊意义，所以这里需要单引号包裹

Figure 3: 在 missing 文件夹中新建一个文件 semester

```

peng-peisu@pengpeisu:~/tmp/missing$ echo '#!/bin/sh' > semester
peng-peisu@pengpeisu:~/tmp/missing$ echo curl --head --silent https://missing.csail.mit.edu >> semester
peng-peisu@pengpeisu:~/tmp/missing$ cat semester
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu

```

Figure 4: 向 semester 中写入内容

4、实例 04：尝试执行这个文件。如果程序无法执行，请使用 **ls** 命令来获取信息并理解其不能执行的原因

- (1) 先执行 **semester** 文件
- (2) 发现不能执行后，输入 **ls -l** 查看是什么原因⁵

(3) 显示 -rw-rw-r- , 表示所有者的权限是可读、可写，但不可执行；组用户权限是可读可写不可执行；其他用户权限是仅可读。³

³ r: 可读 w: 可写 x: 可执行 -: 不可以

```
peng-peisu@pengpeisu:~/tmp/missing$ ./semester
bash: ./semester: Permission denied
peng-peisu@pengpeisu:~/tmp/missing$ ls -l
total 4
-rw-rw-r-- 1 peng-peisu peng-peisu 61 9月 11 14:33 semester
```

Figure 5: 如图所示

5、实例 05：查看 chmod 的手册

- (1) 输入命令：man chmod
- (2) 了解 chmod 的功能：修改文件或目录的权限⁶

```
peng-peisu@pengpeisu: ~
User Commands
CHMOD(1)

NAME
    chmod - change file mode bits

SYNOPSIS
    chmod [OPTION]... MODE[.MODE]... FILE...
    chmod [OPTION]... OCTAL-MODE FILE...
    chmod [OPTION]... --reference=FILE FILE...

DESCRIPTION
    This manual page documents the GNU version of chmod. chmod changes the
    file mode bits of each given file according to mode, which can be ei-
    ther a symbolic representation of changes to make, or an octal number
    representing the bit pattern for the new mode bits.

    The format of a symbolic mode is [ugoa...][[-+=][perms...]..., where
    perms is either zero or more letters from the set rwxXst, or a single
    letter from the set ugo. Multiple symbolic modes can be given, sepa-
    rated by commas.

    A combination of the letters ugoa controls which users' access to the
    file will be changed: the user who owns it (u), other users in the
    Manual page chmod(1) line 1 (press h for help or q to quit)
```

Figure 6: 如图所示

6、实例 06：使用 chmod 命令改变权限，使 semester 文件能够成功执行

- (1) 使用 chmod 命令改变权限⁴
- (2) 再次执行 semester 文件，可以看见已经成功执行
- (3) 输入 ls -l 获取信息可以得知此时用户的权限均为可读可写可执行了⁷

⁴ chmod +x semester 可以为所有用户添加该文件的可执行权限

```
peng-peisu@pengpeisu:~/tmp/missing$ chmod +x semester
peng-peisu@pengpeisu:~/tmp/missing$ ls -l
ls: command not found
peng-peisu@pengpeisu:~/tmp/missing$ ./semester
./semester: 2: curl: not found
peng-peisu@pengpeisu:~/tmp/missing$ ls -l
total 4
-rwxrwxr-x 1 peng-peisu peng-peisu 61 9月 11 14:33 semester
```

Figure 7: 如图所示

7、实例 07: 使用 | 和 > , 将 **semester** 文件输出的最后更改日期信息，写入主目录下的 **last-modified.txt** 的文件中

- (1) 由于虚拟机先前没有安装 curl, 需要安装 curl⁵
- (2) 使用命令: `./semester | grep last-modified > ~/last-modified.txt`, 向文件 `last-modified.txt` 写入信息
- (3) 使用 cat 检验文件中是否已经成功写入了信息⁹

```

peng-peisu@pengpeisu:~/tmp/missing
Get:16 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-backports/universe amd64
DEP-11 Metadata [26.0 kB]
Get:17 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-backports/multiverse amd
64 DEP-11 Metadata [212 B]
Fetched 6,684 kB in 10s (655 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
531 packages can be upgraded. Run 'apt list --upgradable' to see them.
peng-peisu@pengpeisu:~$ sudo apt install curl -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl
The following packages will be upgraded:
  libcurl4
1 upgraded, 1 newly installed, 0 to remove and 530 not upgraded.
Need to get 483 kB of archives.
After this operation, 461 kB of additional disk space will be used.
Get:1 http://mirrors.tuna.tsinghua.edu.cn/ubuntu jammy-updates/main amd64 libcurl4
  amd64 7.81.0-1ubuntu1.20 [289 kB]

```

Figure 8: 成功安装 curl

```

peng-peisu@pengpeisu:~/tmp/missing$ ./semester | grep last-modified > ~/last-modified.txt
peng-peisu@pengpeisu:~/tmp/missing$ cat ~/last-modified.txt
last-modified: Thu, 28 Aug 2025 13:37:00 GMT

```

Figure 9: 如图所示

8、实例 08: 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（例如，使用 **454M** 而不是 **454279954**）；文件以最近修改顺序排序；以彩色文本显示输出结果

- (1) 学习了解 ls 命令的选项: **-a** 显示所有文件（包括隐藏文件）；**-h** 以人类易读的形式显示文件大小；**--sort=time** 使文件以最近修改顺序排列；**--color=auto** 以彩色文本显示输出结果；**-l** 以详细格式显示文件信息⁶ (2) 可以看到对应结果¹⁰

⁶ 命令: `ls -alh --sort=time --color=auto`

```

peng-peisu@pengpeisu:~/tmp/missing$ ls -alh --sort=time --color=auto
total 12K
-rwxrwxr-x 1 peng-peisu peng-peisu 61 9月 11 14:33 semester
drwxrwxr-x 2 peng-peisu peng-peisu 4.0K 9月 11 14:19 .
drwxrwxr-x 3 peng-peisu peng-peisu 4.0K 9月 11 14:17 ..

```

Figure 10: 如图所示

9、实例 09：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

- (1) 创建一个 marco.sh 文件，单独存储函数，以便 debug
- (2) 用 vim 打开 marco.sh 文件，向其中写入函数，如图所示¹¹

```

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

```

Figure 11: 写入函数后，保存并退出

(3) 尝试运行 marco.sh 后发现不能执行，检查发现是不具有可执行的权限。使用 chmod 添加可执行权限

(4) 再次使用 ls -l 检查后发现可执行，输入 marco，可见已经将当前工作目录保存好了

(5) 进入主目录后输入 polo，回到了先前执行 marco 的目录¹²

```

peng-peisu@pengpeisu:~/tmp/missing$ cd tmp/missing
peng-peisu@pengpeisu:~/tmp/missing$ touch marco.sh
peng-peisu@pengpeisu:~/tmp/missing$ vim marco.sh
peng-peisu@pengpeisu:~/tmp/missing$ ls
marco.sh semester
peng-peisu@pengpeisu:~/tmp/missing$ source marco.sh
peng-peisu@pengpeisu:~/tmp/missing$ ls -l
total 8
-rw-rw-r-- 1 peng-peisu peng-peisu 153 9月 12 00:15 marco.sh
-rwxrwxr-x 1 peng-peisu peng-peisu 61 9月 11 14:33 semester
peng-peisu@pengpeisu:~/tmp/missing$ chmod +x marco.sh
peng-peisu@pengpeisu:~/tmp/missing$ source marco.sh
peng-peisu@pengpeisu:~/tmp/missing$ ls -l
total 8
-rwxrwxr-x 1 peng-peisu peng-peisu 153 9月 12 00:15 marco.sh
-rwxrwxr-x 1 peng-peisu peng-peisu 61 9月 11 14:33 semester
peng-peisu@pengpeisu:~/tmp/missing$ marco
save pwd /home/peng-peisu/tmp/missing
peng-peisu@pengpeisu:~/tmp/missing$ cd
peng-peisu@pengpeisu:~$ polo
peng-peisu@pengpeisu:~/tmp/missing$ 

```

Figure 12: 过程如图所示

10、实例 10：编写一段 **bash** 脚本，运行如下的脚本**13**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

```
#!/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"
```

Figure 13: 目标脚本

(1) 先创建一个文件用于存储 **debug** 的脚本**14**

(2) 向其中写入 **debug** 脚本，保存后退出

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

Figure 14: debug 脚本如图所示

(3) 调整该文件的可执行权限，然后执行，可看见在运行 0 次后失败**15**

```
peng-peisu@pengpeisu:~/tmp/missing$ touch debug.sh
peng-peisu@pengpeisu:~/tmp/missing$ vim debug.sh
peng-peisu@pengpeisu:~/tmp/missing$ chmod +x debug.sh
peng-peisu@pengpeisu:~/tmp/missing$ ./debug.sh
failed after 0 times
peng-peisu@pengpeisu:~/tmp/missing$ cat out.log | grep Everything | wc -l
0
```

Figure 15: 过程如图所示

1.2 vim 编辑器

1、实例 11：完成 **vimtutor**。备注：它在一个 **80x24** (80 列, 24 行) 终端窗口看起来效果最好

(1) 回到主目录，输入命令：vimtutor¹⁶，结果如图所示¹⁷

```
peng-peisu@pengpeisu:~/tmp/missing$ cd
peng-peisu@pengpeisu: $ vimtutor
```

Figure 16: 过程如图所示

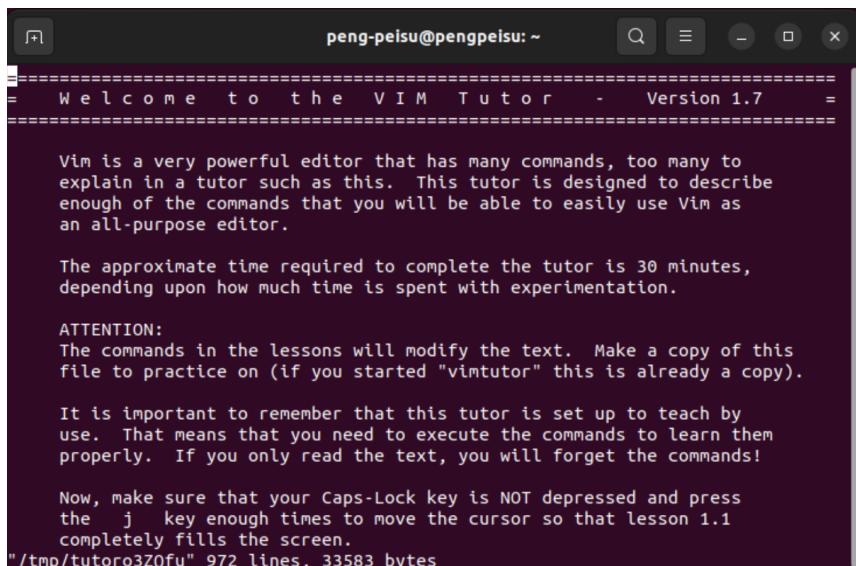


Figure 17: vimtutor

2、实例 12：下载我们的 **vimrc**，然后把它保存到 **~/.vimrc**。

- (1) 使用 curl 命令和课程提供的文件链接将文件保存到 **~/.vimrc**¹⁸
- (2) 通读注释，发现变化：该文件只能用 h/j/k/l 移动，语法高亮等¹⁹

```
peng-peisu@pengpeisu:~/tmp/missing$ curl -o ~/.vimrc https://missing-semester-cn.github.io/2020/files/vimrc
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload Total   Spent   Left Speed
100  3254  100  3254    0     0  1165      0  0:00:02  0:00:02  --- 1165
```

Figure 18: 保存课程文件

3、实例 13：安装和配置一个插件：**ctrlp.vim**

- (1) 用 **mkdir -p ~.vim/pack/vendor/start** 创建插件文件夹
- (2) 下载这个插件：**cd ~.vim/pack/vendor/start; git clone https://github.com/ctrlpvim/ctrlp.vim**²⁰
- (3) 下载后需要在 **~/.vimrc** 中添加如下设置：**set runtimepath=~.vim/pack/vendor/start/ctrlp.vim**²¹
- (4) 打开 vim，输入**:CtrlP**，可看见 **ctrlp.vim** 插件已经开始工作²²

```

peng-peisu@pengpeisu: ~/tmp/missing
6 " Comments in Vimscript start with a '''.
5
4 " If you open this file in Vim, it'll be syntax highlighted for you.
3
2 " Vim is based on Vi. Setting 'nocompatible' switches from the default
1 " Vi-compatibility mode and enables useful Vim functionality. This
7 " configuration option turns out not to be necessary for the file named
1 " '~/.vimrc', because Vim automatically enters nocompatible mode if that fil
e
2 " is present. But we're including it here just in case this config file is
3 " loaded some other way (e.g. saved as 'foo', and then Vim started with
4 " 'vim -u foo').
5 set nocompatible
6
7 " Turn on syntax highlighting.
8 syntax on
9
10 " Disable the default Vim startup message.
11 set shortmess+=I
12
13 " Show line numbers.
14 set number
./.vimrc [R0]                                7,1          Top

```

Figure 19: 该课程文件内容

```

peng-peisu@pengpeisu:~$ cd tmp/missing
peng-peisu@pengpeisu:~/tmp/missing$ cd ~/.vim/pack/vendor/start
peng-peisu@pengpeisu:~/.vim/pack/vendor/start$ git clone https://github.com/ctrl
pvim/ctrlp.vim
Cloning into 'ctrlp.vim'...
remote: Enumerating objects: 4316, done.
remote: Counting objects: 100% (185/185), done.
remote: Compressing objects: 100% (117/117), done.
remote: Total 4316 (delta 78), reused 162 (delta 67), pack-reused 4131 (from 1)
Receiving objects: 100% (4316/4316), 1.71 MiB | 2.39 MiB/s, done.
Resolving deltas: 100% (1668/1668), done.

```

Figure 20: 下载插件过程

```

3
2 " Show line numbers.
1 set number
22 set runtimepath^=~/vim/pack/vendor/start/ctrlp.vim
1 " This enables relative line numbering mode. With both number and
2 " relativenumber enabled, the current line shows the true line number, while
3 " all other lines (above and below) are numbered relative to the current lin

```

Figure 21: 在 .vimrc 中添加内容

```

[No Name]                                         0,0-1          All
> 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 <mru>=<files>=<buf> <-> /home/peng-peisu/.vim/pack/vendor/start

```

Figure 22: 可见插件已起作用

4、实例 14：自定义 CtrlP：添加 configuration 到你的 `/.vimrc` 来用按 **Ctrl-P** 打开 **CtrlP**

- (1) 向 `.vimrc` 文件的末尾添加命令`23`, 将`:CtrlP` 的快捷键设置为 `ctrl+p`
 - (2) 保存并退出后打开 vim, 直接按下快捷键, 可以看到插件运行

Figure 23: 添加内容如图末尾

5、实例 15：进一步自定义你的 `/.vimrc` 和安装更多插件

- (1) 安装 Vim 的包管理器 vim-plug，来安装插件²⁴
 - (2) 在`~/.vimrc` 中添加自己需要的插件²⁵
 - (3) 在 vim 命令行中输入`:PlugInstall` 并回车，执行安装插件²⁶

```
peng-peisu@pengpeisu:~/tmp/missing$ curl -fLo ~/.vim/autoload/plug.vim --create-  
dirs \  
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim  
% Total    % Received % Xferd  Average Speed   Time   Time  Current  
          Dload  Upload Total Spent   Left Speed  
100 84862  100 84862    0     0  31530      0  0:00:02  0:00:02  ----- 31523
```

Figure 24: 安装包管理器 vim-plug

```
4 call plug#begin()  
3 Plug 'preservim/NERDTree'  
2 call plug#end()  
1
```

Figure 25: 修改`~/.vimrc`文件

```
[Plugins] [No Name]
1 Installing plugins (0/1)
2 + NERDTree: Cloning into '/home/peng-peisu/.vim/plugged/NERDTree'...
[Plugins]
:PlugInstall          2,1      All
```

Figure 26: 执行安装命令

1.3 数据整理

1、实例 16：统计 words 文件 (**/usr/share/dict/words**) 中包含至少三个 a 且不以's 结尾的单词个数

- (1) 在主目录输入命令
- (2) 输出单词个数: 854²⁷

```
peng-peisu@pengpeisu:~$ cat /usr/share/dict/words | tr "[upper:]" "[lower:]" |
grep -E "^([^a]*a){3}.*$" | grep -v "'s$" | wc -l
854
```

Figure 27: 如图所示

2、实例 17：这些单词中，出现频率前三的末尾两个字母是什么？

- (1) 输入命令
- (2) 根据输出结果可以看出出现频率前三的末尾两个字母分别是 ca、ce、am²⁸

```
peng-peisu@pengpeisu:~$ cat /usr/share/dict/words | tr "[upper:]" "[lower:]" |
grep -E "^([^a]*a){3}.*$" | grep -v "'s$" | sed -E "s/.*( [a-z]{2})$/\1/" | sort |
uniq -c | sort | tail -n3
     8 am
     8 ce
     9 ca
```

Figure 28: 如图所示

3、实例 18：共存在多少种词尾两字母组合？

- (1) 输入命令
- (2) 有输出可知共有 112 种词尾两字母组合²⁹

```
peng-petsu@pengpetsu:~$ cat /usr/share/dict/words | tr "[upper:]" "[lower:]" | grep -E "^(aa){3}.*$" | grep -v "s$" | sed -E "s/.*(aa){2})$/\1/" | sort | uniq | wc -l
112
```

Figure 29: 如图所示

4、实例 19: 进行原地替换听上去很有诱惑力, 例如:**sed s/REGEX/SUBSTITUTION/**
input.txt > input.txt。但是这并不是一个明智的做法, 为什么呢? 还是说只有 **sed** 是这样的?

- (1) 执行命令 > 时会先触发重定向功能, 清空 **input.txt** 的内容, 之后才会执行 **sed**。这会导致数据彻底被清除, 不可逆。
- (2) 不是 **sed** 才这样。关键在于命令 >。所有依赖于 > 重定向的命令都会有这个问题。

5、实例 20: 查询该主机的开机记录

- (1) 输入命令 **last reboot** 查询开机记录
- (2) 输入命令提取开机时间[30](#)

```
peng-petsu@pengpetsu:~$ last reboot
reboot    system boot  6.8.0-79-generic Fri Sep 12 09:46   still running
reboot    system boot  6.8.0-79-generic Thu Sep 11 14:11 - 09:45  (19:34)

wtmp begins Thu Sep 11 14:11:27 2025
peng-petsu@pengpetsu:~$ last reboot | awk 'NR<=11 && $1=="reboot"{print $3}' | tail -n +2
boot
```

Figure 30: 如图所示

2

解题感悟

2.1 Shell 工具和脚本

- 1、理解到了 **Shell** 的高效率。通过脚本就可以批量地完成重复操作，大大提高了用户的效率。
- 2、理解到了 **Shell** 可组合性下带来的便利。通过各种命令的组合就可以一次性完成复杂的任务。
- 3、在学习 **Shell** 工具与脚本的过程中，了解到了更多常用的命令及其功能，也实践了其中的诸多操作，给我带了对命令行的更深的理解。

2.2 vim 编辑器

- 1、首先，我加深了对 **vim** 基础操作的印象。比如除了移动鼠标外，**h/j/k/l** 控制上下左右的方式也更加熟练。还有进入模式切换、退出 **vim** 的方式等基础操作。
- 2、其次，对于 **vim** 相关插件如何安装也有了了解。比如包管理器 **vim-plug** 在插件安装中起到的作用。

2.3 数据整理

- 1、体会到了 **Shell** 和 **vim** 在数据整理上起到的作用。通过一句命令就可以轻松对一个文件的数据进行遍历、整理、计算、输出。