

第二次实验报告

宫怡

学号：24020007033

2025.9.12

目录

1	Shell 工具和脚本	2
1.1	检查 shell 是否满足要求	2
1.2	创建文件夹	2
1.3	用 man 查看 touch 使用手册	2
1.4	用 touch 新建文件	3
1.5	写入文件	3
1.6	执行文件	3
1.7	日期信息写入文件	3
1.8	ls 使用	4
1.9	编写函数	4
1.10	重现错误并捕获输出	5
1.11	查找并压缩 HTML 文件	6
2	编辑器 vim	7
2.1	vimtutor	7
2.2	下载保存 vimrc	7
2.3	安装配置 ctrlp.vim	8
2.4	重做演示	8
2.5	转换文件	10
3	数据整理	11
3.1	words 文件统计	11
3.2	原地替换	11
3.3	查找开机信息	11
3.4	统计开机时间	12
4	心得体会	13

实验内容: 1.Shell 工具和脚本

2. 编辑器 (Vim)
3. 数据整理

1 Shell 工具和脚本

1.1 检查 shell 是否满足要求

```
gy@gy-virtual-machine:~/Desktop$ echo $SHELL
/bin/bash
```

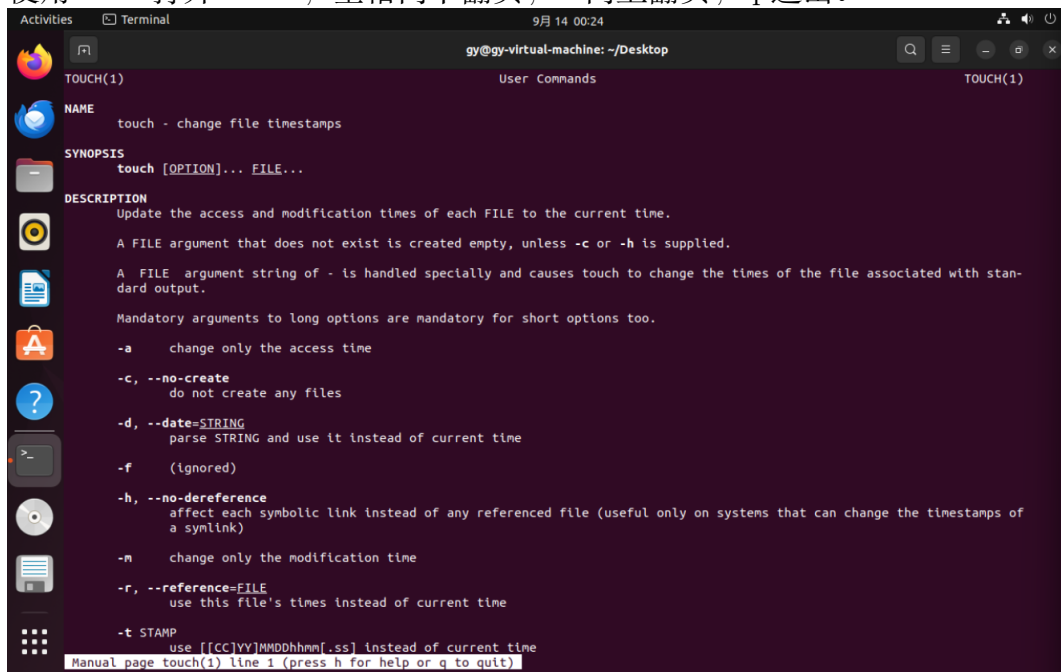
使用 “echo \$SHELL” 检查 Shell 是否符合标准, 返回/bin/bash, 说明 shell 符合标准

1.2 创建文件夹

使用 “mkdir /tmp/missing” 创建文件夹 missing。

1.3 用 man 查看 touch 使用手册

使用 man 打开 touch, 空格向下翻页, b 向上翻页, q 退出。



1.4 用 touch 新建文件

“touch /tmp/missing/semester” 用 touch 新建文件 semester。

1.5 写入文件

Listing 1: 写入文件

```
1 echo '#!/bin/sh' > /tmp/missing/semester
2 echo 'curl --head --silent https://missing.csail.mit.edu' >> /
  tmp/missing/semester
```

第一行用 > 重定向覆盖写入（创建并写入!/bin/sh）。

第二行用 » 追加写入。

1.6 执行文件

使用 “./semester” 运行文件，使用 “ls -l semester” 显示错误信息，给文件添加执行权限，再次执行。

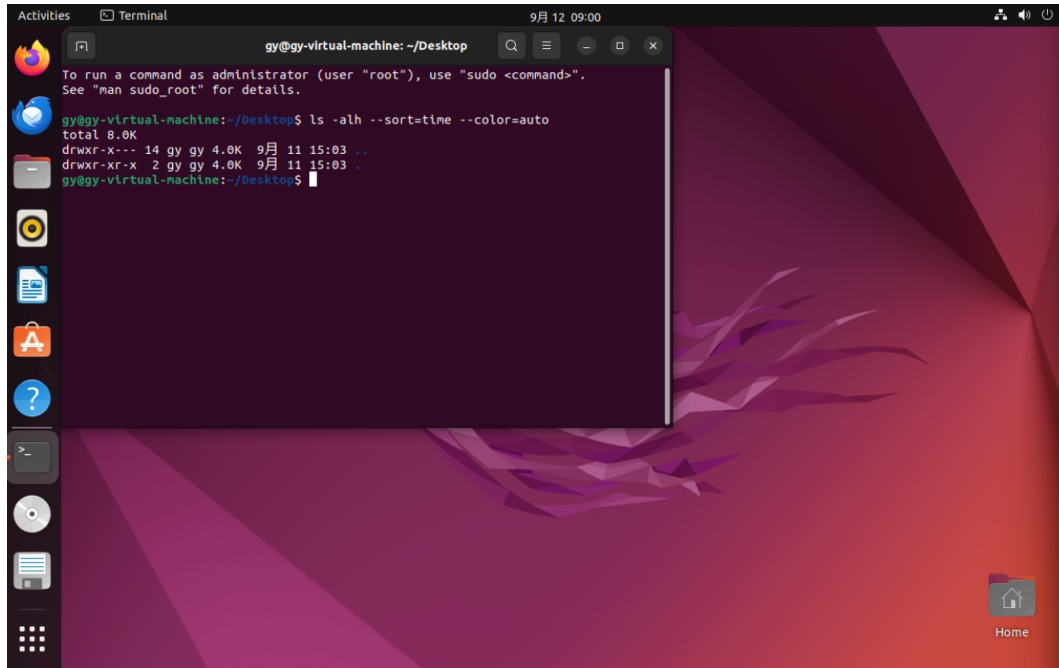
```
gy@gy-virtual-machine:~/Desktop$ touch /tmp/missing/semester
gy@gy-virtual-machine:~/Desktop$ echo '#!/bin/sh' > /tmp/missing/semester
gy@gy-virtual-machine:~/Desktop$ echo 'curl --head --silent https://missing.csail.mit.edu' >> /tmp/missing/semester
gy@gy-virtual-machine:~/Desktop$ man chmod
gy@gy-virtual-machine:~/Desktop$ cd /tmp/missing
gy@gy-virtual-machine:~/tmp/missing$ ./semester
bash: ./semester: Permission denied
gy@gy-virtual-machine:~/tmp/missing$ ls -l semester
-rw-rw-r-- 1 gy gy 61  9月 14 00:29 semester
gy@gy-virtual-machine:~/tmp/missing$ chmod +x semester
gy@gy-virtual-machine:~/tmp/missing$ ./semester
HTTP/2 200
server: GitHub.com
content-type: text/html; charset=utf-8
last-modified: Thu, 28 Aug 2025 13:37:00 GMT
access-control-allow-origin: *
etag: "68b05b7c-2002"
expires: Sat, 13 Sep 2025 05:08:32 GMT
cache-control: max-age=600
x-proxy-cache: MISS
x-github-request-id: 709C:149EB3:88DFF:93250:68C4F9F8
accept-ranges: bytes
age: 0
date: Sun, 14 Sep 2025 02:42:02 GMT
via: 1.1 varnish
x-served-by: cache-sin-wsss1830072-SIN
x-cache: HIT
x-cache-hits: 0
x-timer: S1757817722.030499,V0,V0,VE239
vary: Accept-Encoding
x-fastly-request-id: 452e3aa273d19d7d019dd8977ebbcc4bbcc3f67f8
content-length: 8194
```

1.7 日期信息写入文件

使用 grep 将信息写入文件，使用 cat 检查写入的信息。

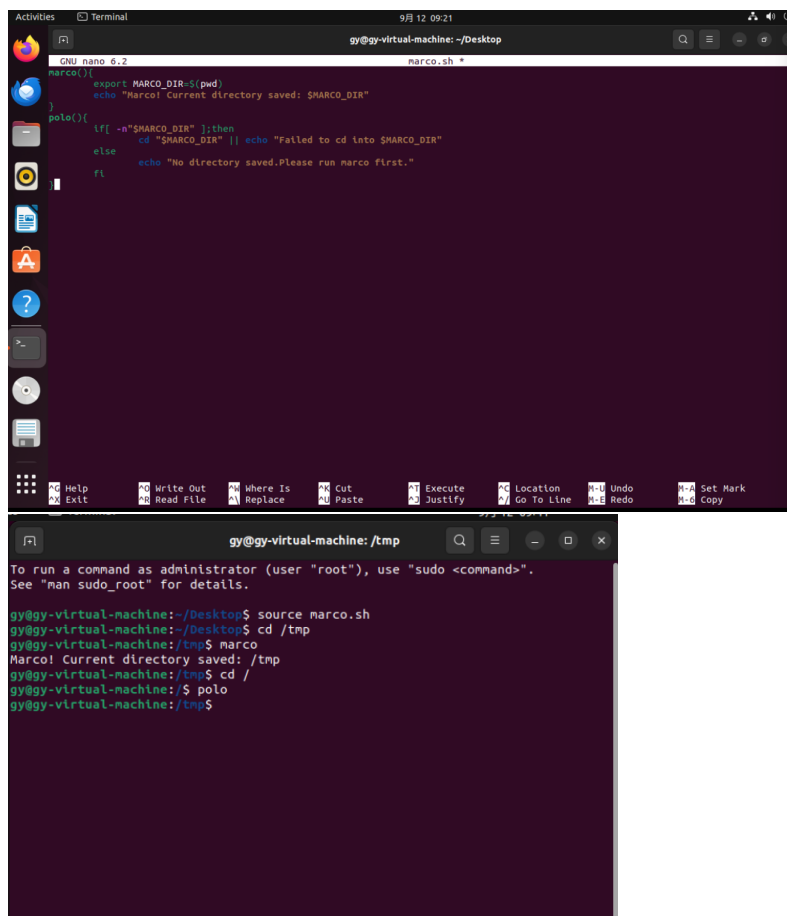
```
gy@gy-virtual-machine:/tmp/missing$ /tmp/missing/semester | grep 'last-modified' > ~/last-modified.txt
gy@gy-virtual-machine:/tmp/missing$ cat ~/last-modified.txt
last-modified: Thu, 28 Aug 2025 13:37:00 GMT
```

1.8 ls 使用



1.9 编写函数

使用 nano 命令创建 marco.sh 文件，在其中编写 marco, polo 函数，通过 source 命令加载函数，使用 marco 的时候，文件会保存当前目录，使用 polo 返回到上次保存的目录。

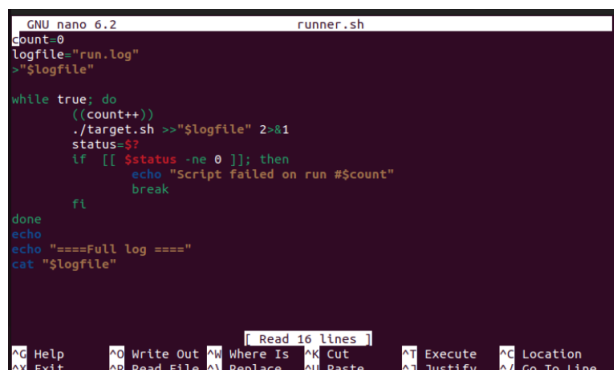


The screenshot shows a terminal window with two panes. The top pane shows the creation of a shell script named `marco.sh` using `nano`. The script defines a function `marco()` that sets `MARCO_DIR` to the current directory and prints it. It also defines a function `polo()` that checks if `MARCO_DIR` is set and either changes to that directory or prints an error message. The bottom pane shows the execution of the script. The user sources the script, changes to the `/tmp` directory, runs `marco` (which prints `/tmp`), and then runs `polo` (which prints `No directory saved. Please run marco first.`).

```
GNU nano 6.2 marco.sh
marco(){
    export MARCO_DIR=$(pwd)
    echo "Marco! Current directory saved: $MARCO_DIR"
}
polo(){
    if [ -n "$MARCO_DIR" ]; then
        cd "$MARCO_DIR" || echo "Failed to cd into $MARCO_DIR"
    else
        echo "No directory saved. Please run marco first."
    fi
}

gy@gy-virtual-machine: ~/Desktop
gy@gy-virtual-machine:~/Desktop$ source marco.sh
gy@gy-virtual-machine:~/Desktop$ cd /tmp
gy@gy-virtual-machine: /tmp$ marco
Marco! Current directory saved: /tmp
gy@gy-virtual-machine: /tmp$ cd /
gy@gy-virtual-machine:/$ $ polo
gy@gy-virtual-machine: /tmp$
```

1.10 重现错误并捕获输出



The screenshot shows a terminal window with a single pane displaying the contents of a shell script named `runner.sh`. The script sets `count` to 0, defines `logfile` as `run.log`, and opens the file for writing. It then enters a `while true` loop that increments `count`, runs `./target.sh` with output redirected to `$logfile`, and checks the status. If the status is non-zero, it prints an error message and breaks the loop. After the loop, it prints `====Full log====` and displays the contents of `$logfile`.

```
GNU nano 6.2 runner.sh
count=0
logfile="run.log"
>"$logfile"

while true; do
    ((count++))
    ./target.sh >>"$logfile" 2>&1
    status=$?
    if [[ $status -ne 0 ]]; then
        echo "Script failed on run # $count"
        break
    fi
done
echo "====Full log===="
cat "$logfile"

[ Read 16 lines ]
```

```
gy@gy-virtual-machine:~/Desktop$ nano target.sh
gy@gy-virtual-machine:~/Desktop$ ./runner.sh
Script failed on run #17

====Full log====
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
Everything went according to plan
Something went wrong
The error was using magic numbers
```

编写 runner.sh，通过记录 target.sh 每一次的运行状态，记录成功运行的次数，直到捕获到错误。

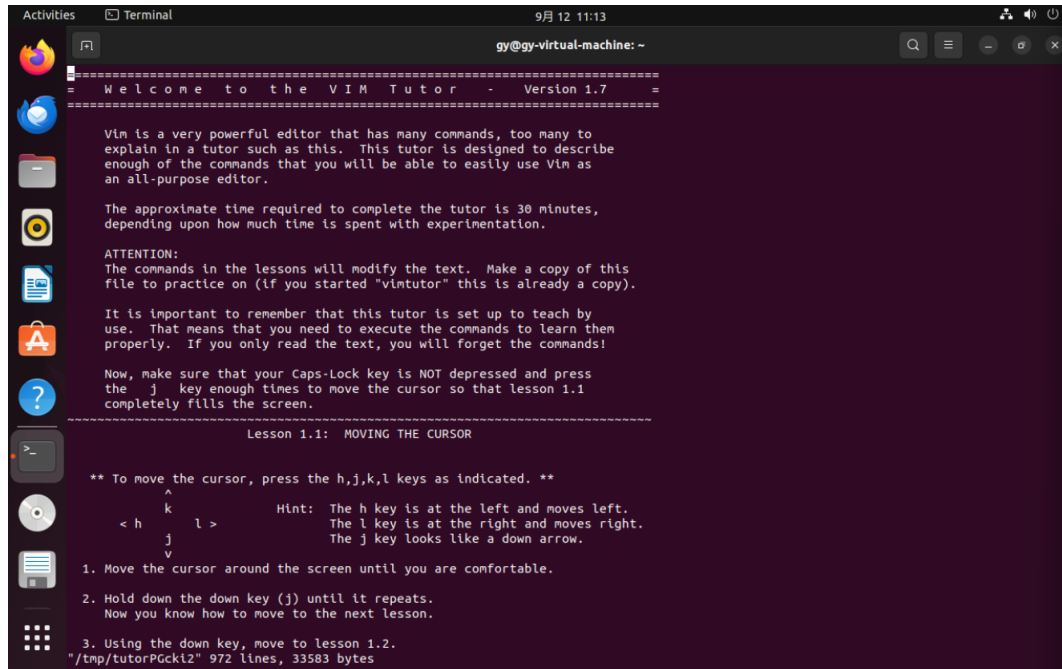
1.11 查找并压缩 HTML 文件

创建 test1_、test2_、test3.html 文件用于测试，运行压缩命令“find . -name '*.html' -print0 | xargs -0 zip html_files.zip”，通过 find. 进行递归查找，利用-name '*.html' 匹配后缀为.html 的文件，xargs -0 以 NUL 为分隔符读取输入，把文件名作为参数传递给后面的命令。

```
gy@gy-virtual-machine:~$ find . -name "*.html"
gy@gy-virtual-machine:~$ echo "<h1>hello</h1>" >test1.html
gy@gy-virtual-machine:~$ echo "<p> World</p>" >test2.html
gy@gy-virtual-machine:~$ mkdir subdir
gy@gy-virtual-machine:~$ echo "<div>Subdir</div>" > subdir/test3.html
gy@gy-virtual-machine:~$ find . -name "*.html" -print0 | xargs -0 zip html_files.zip
adding: test2.html (stored 0%)
adding: test1.html (stored 0%)
adding: subdir/test3.html (deflated 6%)
```

2 编辑器 vim

2.1 vimtutor



```
gy@gy-virtual-machine: ~
=====
= 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 "vimtutor" 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 then
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. **

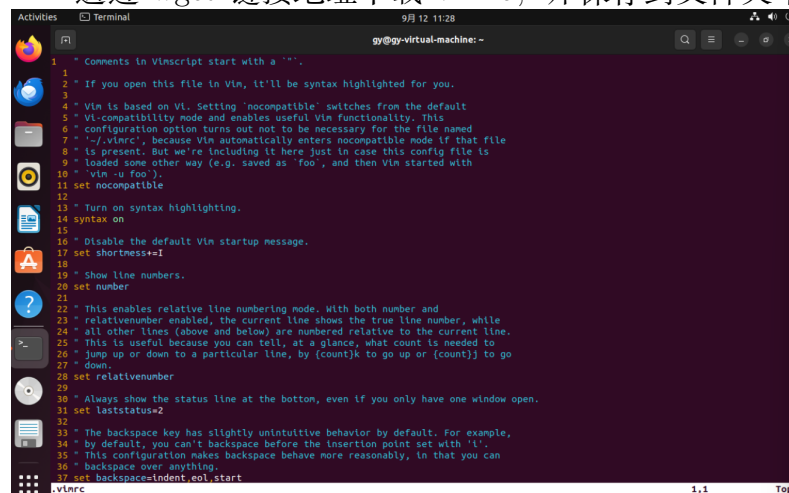
      ^
      k
< h   l >      Hint: The h key is at the left and moves left.
      j          The l key is at the right and moves right.
      v          The j key looks like a down arrow.

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.
"/tmp/tutorPGck12" 972 lines, 33583 bytes
```

sudo apt install vim 安装 vim，使用 vimtutor 进入 vimtutor，学习使用 vim

2.2 下载保存 vimrc

通过 wget 链接地址下载 vimrc，并保存到文件夹中，然后打开阅读 vimrc



```
gy@gy-virtual-machine: ~
1  " Comments in Vimscript start with a " ".
2  " If you open this file in Vim, it'll be syntax highlighted for you.
3
4  " Vim is based on Vi. Setting 'nocompatible' switches from the default
5  " Vi-compatibility mode and enables useful Vim functionality. This
6  " configuration option turns out not to be necessary for the file named
7  " '~/.vimrc', because Vim automatically enters nocompatible mode if that file
8  " is present. But we're including it here just in case this config file is
9  " loaded some other way (e.g. saved as 'foo', and then Vim started with
10 " 'vim -u foo').
11 set nocompatible
12
13 " Turn on syntax highlighting.
14 syntax on
15
16 " Disable the default Vim startup message.
17 set shortmess+=I
18
19 " Show line numbers.
20 set number
21
22 " This enables relative line numbering mode. With both number and
23 " relativenumber enabled, the current line shows the true line number, while
24 " all other lines (above and below) are numbered relative to the current line.
25 " This is useful because you can tell, at a glance, what count is needed to
26 " jump up or down to a particular line, by [count]k to go up or [count]j to go
27 " down.
28 set relativenumber
29
30 " Always show the status line at the bottom, even if you only have one window open.
31 set laststatus=2
32
33 " The backspace key has slightly unintuitive behavior by default. For example,
34 " by default, you can't backspace before the insertion point set with 'i'.
35 " This configuration makes backspace behave more reasonably, in that you can
36 " backspace over anything.
37 set backspace=indent,eol,start
38
39 .vimrc
40 1.1 Top
```



```

gy@gy-virtual-machine:~$ wget https://missing-semester-cn.github.io/2020/files/vimrc -O ~/.vimrc
--2025-09-12 11:27:00-- https://missing-semester-cn.github.io/2020/files/vimrc
Resolving missing-semester-cn.github.io (missing-semester-cn.github.io)... 185.199.108.153, 185.199.109.153, 185.199.110.153, ...
Connecting to missing-semester-cn.github.io (missing-semester-cn.github.io)[185.199.108.153]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3254 (3.2K) [application/octet-stream]
Saving to: '/home/gy/.vimrc'

/home/gy/.vimrc      100%[=====] 3.18K --.-KB/s  in 0s
2025-09-12 11:27:16 (13.1 MB/s) - '/home/gy/.vimrc' saved [3254/3254]

gy@gy-virtual-machine:~$ vim ~/.vimrc

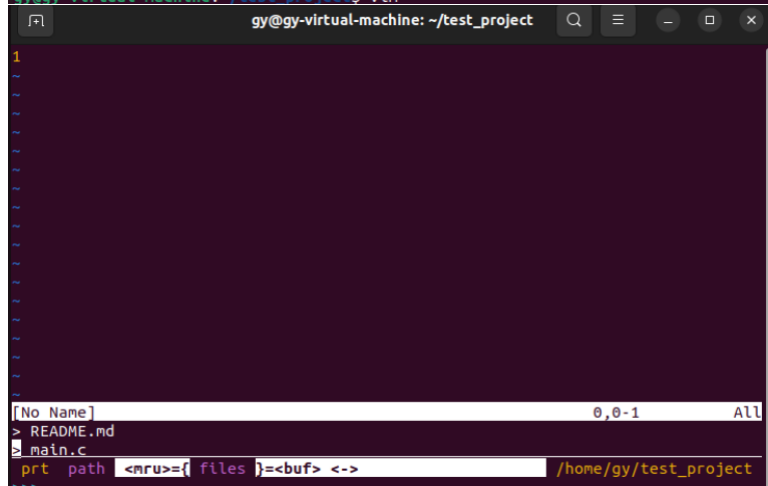
```

2.3 安装配置 ctrlp.vim

```

gy@gy-virtual-machine:~$ cd ~/.vim/pack/vendor/start
gy@gy-virtual-machine:~/.vim/pack/vendor/start$ git clone https://github.com/ctrlpvim/ctrlp.vim
Cloning into 'ctrlp.vim'...
remote: Enumerating objects: 4316, done.
remote: Counting objects: 100% (185/185), done.
remote: Compressing objects: 100% (113/113), done.
remote: Total 4316 (delta 78), reused 164 (delta 71), pack-reused 4131 (from 1)
Receiving objects: 100% (4316/4316), 1.71 MiB | 104.00 KiB/s, done.
Resolving deltas: 100% (1668/1668), done.
gy@gy-virtual-machine:~/.vim/pack/vendor/start$ mkdir -p ~/test_project
gy@gy-virtual-machine:~/.vim/pack/vendor/start$ cd ~/test_project
gy@gy-virtual-machine:~/test_project$ echo "int main(){return 0;}" >main.c
gy@gy-virtual-machine:~/test_project$ echo "# utils" > README.md
gy@gy-virtual-machine:~/test_project$ vim

```



创建插件目录, 使用

git 下载插件, 新建一个工程目录, 在里面创建 main.c, README.md 文件, 用作练习, 在 vim 里面输入: CtrlP, 就能搜索到 main.c 和 README.md 文件, 输入回车就可以进入到需要的文件。

2.4 重做演示

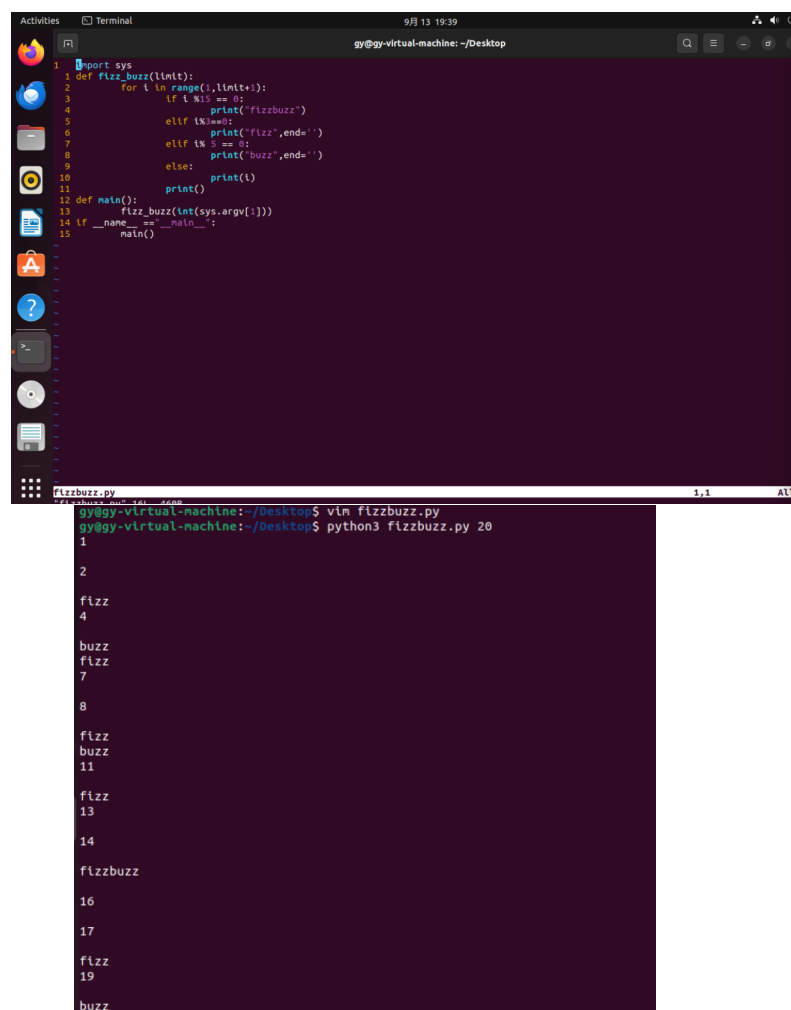
Listing 2: 有问题的 fizzbuzz 实现

```

1 def fizz_buzz(limit):
2     for i in range(limit):

```

```
3         if i % 3 == 0:
4             print('fizz')
5         if i % 5 == 0:
6             print('fizz')
7         if i % 3 and i % 5:
8             print(i)
9
10 def main():
11     fizz_buzz(10)
```



```
gy@gy-virtual-machine: ~/Desktop
1 import sys
2 def fizz_buzz(limit):
3     for i in range(1, limit+1):
4         if i % 15 == 0:
5             print("fizzbuzz")
6         elif i % 3 == 0:
7             print("fizz", end=" ")
8         elif i % 5 == 0:
9             print("buzz", end=" ")
10        else:
11            print(i)
12
13 def main():
14     fizz_buzz(int(sys.argv[1]))
15 if __name__ == "__main__":
16     main()
17
18 gy@gy-virtual-machine: ~/Desktop$ vim fizzbuzz.py
19 gy@gy-virtual-machine: ~/Desktop$ python3 fizzbuzz.py 20
20 1
21 2
22 fizz
23 4
24 buzz
25 fizz
26 7
27 8
28 fizz
29 buzz
30 11
31 12
32 fizz
33 14
34 fizzbuzz
35 16
36 17
37 fizz
38 19
39 buzz
```

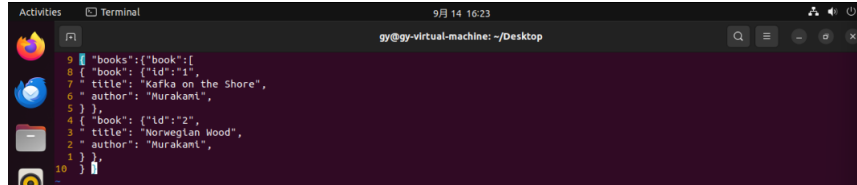
创建新的 `fizzbuzz.py` 文件，通过 “`gg O import sys <Esc>`” 在文件头插入 `import sys`，使用 “`G`” 跳到文件最后一行，“`o`” 在下一行打开新行并进入插入模

式，输入主函数，使用 vim 命令行可以快速找到相应位置进行操作。

2.5 转换文件

使用 vim 宏的命令改变 example.xml 的格式，将其转换为 JSON 文件

```
<?xml version="1.0"?>
<books>
  <book id="1">
    <title>Kafka on the Shore</title>
    <author>Murakami</author>
  </book>
  <book id="2">
    <title>Norwegian Wood</title>
    <author>Murakami</author>
  </book>
</books>
```

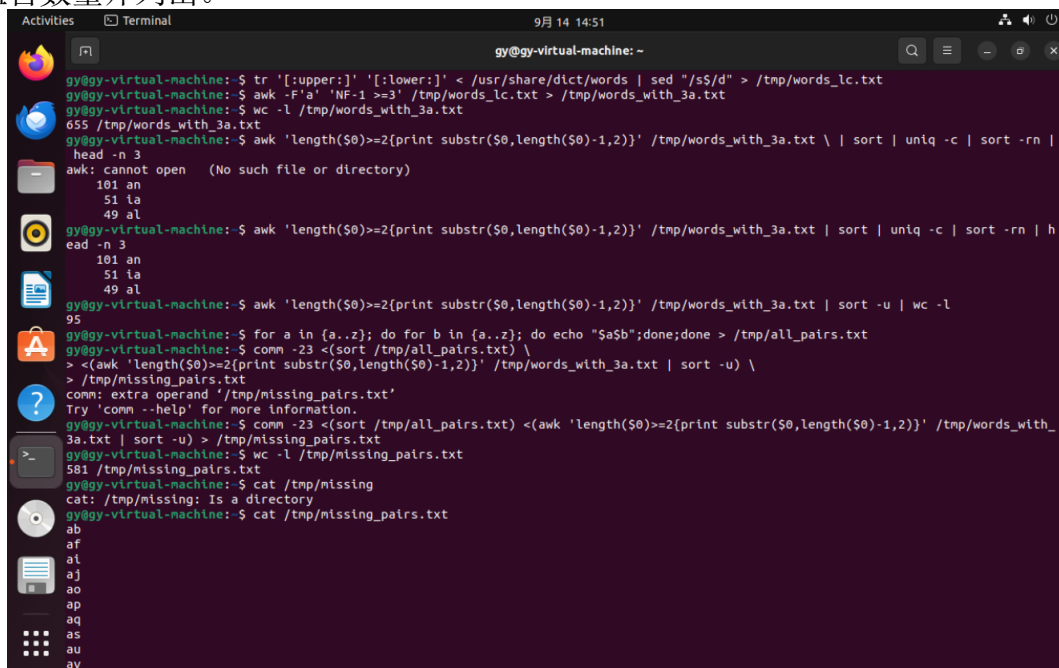


```
9  "books": {"book": [
8  { "book": { "id": "1",
7  "title": "Kafka on the Shore",
6  "author": "Murakami",
5  } },
4  { "book": { "id": "2",
3  "title": "Norwegian Wood",
2  "author": "Murakami",
1  } },
10 ] }
```

3 数据整理

3.1 words 文件统计

用 `tr` 把词表统一转为小写，用 `sed` 去掉以 ‘s’ 结尾的行，并用 `awk` 统计每个单词中的 a 出现次数 (`-F'a'`)。结果会写到/tmp 的临时文件中，而后输出未出现的组合数量并列出。



```
gy@gy-virtual-machine: ~
gy@gy-virtual-machine:~$ tr '[:upper:]' '[:lower:]' < /usr/share/dict/words | sed "/s$/d" > /tmp/words_lc.txt
gy@gy-virtual-machine:~$ awk -F'a' 'NF-1 >=3' /tmp/words_lc.txt > /tmp/words_with_3a.txt
gy@gy-virtual-machine:~$ wc -l /tmp/words_with_3a.txt
655 /tmp/words_with_3a.txt
gy@gy-virtual-machine:~$ awk 'length($0)>=2{print substr($0,length($0)-1,2)}' /tmp/words_with_3a.txt | sort | uniq -c | sort -rn | head -n 3
awk: cannot open (No such file or directory)
101 an
51 ia
49 al
gy@gy-virtual-machine:~$ awk 'length($0)>=2{print substr($0,length($0)-1,2)}' /tmp/words_with_3a.txt | sort | uniq -c | sort -rn | head -n 3
101 an
51 ia
49 al
gy@gy-virtual-machine:~$ awk 'length($0)>=2{print substr($0,length($0)-1,2)}' /tmp/words_with_3a.txt | sort -u | wc -l
95
gy@gy-virtual-machine:~$ for a in {a..z}; do for b in {a..z}; do echo "$a$b";done;done > /tmp/all_pairs.txt
gy@gy-virtual-machine:~$ comm -23 <(sort /tmp/all_pairs.txt) \
> <(awk 'length($0)>=2{print substr($0,length($0)-1,2)}' /tmp/words_with_3a.txt | sort -u) \
> /tmp/missing_pairs.txt
comm: extra operand '/tmp/missing_pairs.txt'
Try 'comm --help' for more information.
gy@gy-virtual-machine:~$ comm -23 <(sort /tmp/all_pairs.txt) <(awk 'length($0)>=2{print substr($0,length($0)-1,2)}' /tmp/words_with_3a.txt | sort -u) > /tmp/missing_pairs.txt
gy@gy-virtual-machine:~$ wc -l /tmp/missing_pairs.txt
581 /tmp/missing_pairs.txt
gy@gy-virtual-machine:~$ cat /tmp/missing
cat: /tmp/missing: Is a directory
gy@gy-virtual-machine:~$ cat /tmp/missing_pairs.txt
ab
af
ai
aj
ao
ap
aq
as
au
av
```

3.2 原地替换

“> input.txt”会在命令执行之前就把 input.txt 截断为 0 字节。当 `sed` 尝试从 input.txt 读取内容时，文件已经空了，所以输出的内容也会是空的。可以使用 “`sed -i 's/REGEX/SUBSTITUTION/' input.txt`”，使用临时文件做来替换原文件。

3.3 查找开机信息

利用 `journalctl` 查看 `systemd` 的日志，通过 `journalctl -list-boots` 列出最近的开机记录，统计开机记录的数量

3.4 统计开机时间

提取开机开始和结束时间，结合 startup 和 finished 计算开机时间

```
gy@gy-virtual-machine:~$ journalctl --list-boots | head -n 10
0 8c8afe03db514537b08b25f1f27d0808a Thu 2025-09-11 15:03:13 CST-Sun 2025-09-14 15:48:06 CST
gy@gy-virtual-machine:~$ journalctl --list-boots | wc -L
1
gy@gy-virtual-machine:~$ journalctl --b 0 | grep -E "systemd\[.*\]:( Startup finished|Starting version)"
9月 11 15:04:06 gy-virtual-machine systemd[1]: Startup finished in 9.681s (kernel) + 53.967s (userspace) = 1min 3.649s.
9月 11 15:04:10 gy-virtual-machine systemd[879]: Startup finished in 32.816s.
gy@gy-virtual-machine:~$ journalctl -b -l |grep "Startup finished"
9月 11 15:04:06 gy-virtual-machine systemd[1]: Startup finished in 9.681s (kernel) + 53.967s (userspace) = 1min 3.649s.
9月 11 15:04:10 gy-virtual-machine systemd[879]: Startup finished in 32.816s.
```

4 心得体会

在这次实验过程中，我主要接触和练习了 Shell 工具、Vim 文本编辑器以及数据整理命令，对命令行操作和文本处理有了更加直观的理解和体会。

通过这次实验，我的总体体会是：Shell 工具负责调度与组合，Vim 负责编辑与微调，数据整理命令负责分析与统计。三者结合，使得文本处理和数据管理的效率大大提高。这种“命令行思维”让我更加意识到工具背后的逻辑：不一定要写复杂的程序，很多时候几条简单的命令就能解决问题。