



中国海洋大学
OCEAN UNIVERSITY OF CHINA

第二周

实 验 报 告

学 院： 信息科学与工程学部

班 号： 计科一班

姓 名： 顾晓宁

学 号： 24020007036

实验编号： 第二周实验报告

指导教师： 周小伟

目录

1 实验目的	4
2 练习内容与结果	4
2.1 bash命令	4
2.1.1 在 /tmp 下新建一个名为 missing 的文件夹?	4
2.1.2 用 man 查看程序 touch 的使用手册?	4
2.1.3 用 touch 在 missing 文件夹中新建一个叫 semester 的文件?	5
2.1.4 将以下内容一行一行地写入 semester 文件:	5
2.1.5 尝试执行这个文件。例如, 将该脚本的路径 (./semester) 输入到您的 shell 中并回车?	5
2.1.6 查看 chmod 的手册(例如, 使用 man chmod 命令)?	6
2.1.7 使用 chmod 命令改变权限, 使 ./semester 能够成功执行, 不要使用 sh semester 来执行该程序。您的 shell 是如何知晓这个文件需要使用 sh 来解析呢?	6
2.1.8 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	7
2.1.9 写一段命令来从 /sys 中获取笔记本的电量信息?	8
2.1.10 阅读 man ls, 然后使用 ls 命令进行如下操作?	8
2.2 bash脚本	9
2.2.1 编写两个 bash 函数 marco 和 polo 执行下面的操作	9
2.2.2 编写一段 bash 脚本, 完成以下任务?	10
2.2.3 我们已经知道, 命令行可以从参数或标准输入接受输入。在用管道连接命令时, 我们将标准输出和标准输入连接起来。任务:	11
2.2.4 编写一个命令或脚本递归的查找文件夹中最近修改的文件。更通用的做法, 你可以按照最近的修改时间列出文件吗?	12
2.2.5 编写一个脚本, 处理用户信息, 演示字符串变量、数组和环境变量的使用?	12
2.3 vim基础操作	13
2.3.1 完成 vimtutor?	13
2.3.2 下载我们的 vimrc, 然后把它保存到 ~/.vimrc。通读这个注释详细的文件 (用 Vim!), 然后观察 Vim 在这个新的设置下看起来和使用起来有哪些细微的区别?	13
2.3.3 下载vim-plug?	14
2.4 数据整理/正则表达式	15
2.4.1 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?	15
2.4.2 这些单词中, 出现频率前三的末尾两个字母是什么?	15
3 实验感悟	16
4 个人github/gitee账号	16

第二周实验：Shell与Shell脚本、vim、数据整理

1 实验目的

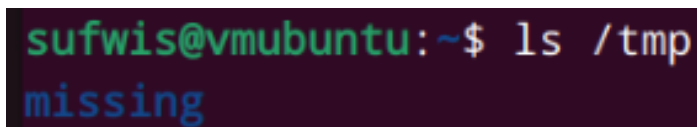
练习使用bash和vim，以及如何编写bash脚本和进行数据整理

2 练习内容与结果

2.1 bash命令

2.1.1 在 /tmp 下新建一个名为 missing 的文件夹？

```
mkdir /tmp/missing
```

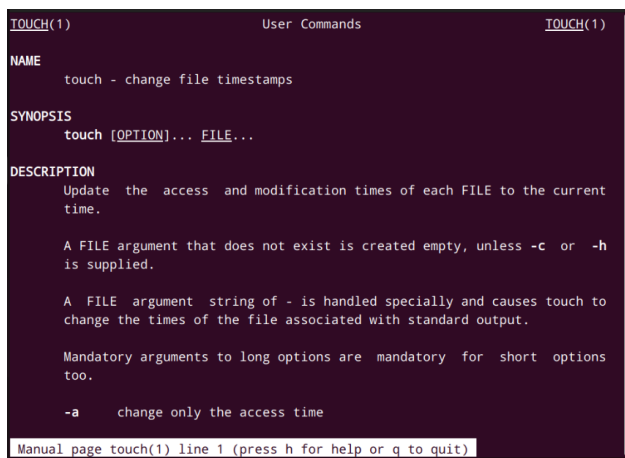


```
sufwis@vmubuntu:~$ ls /tmp
missing
```

图 1: 结果如图

2.1.2 用 man 查看程序 touch 的使用手册？

```
man touch
```



```
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.

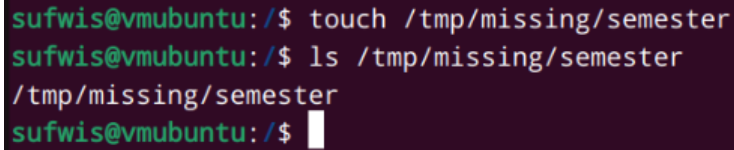
    -a      change only the access time

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

图 2: 如图所示为touch的man手册

2.1.3 用 touch 在 missing 文件夹中新建一个叫 semester 的文件?

```
1 touch /tmp/missing/semester
```



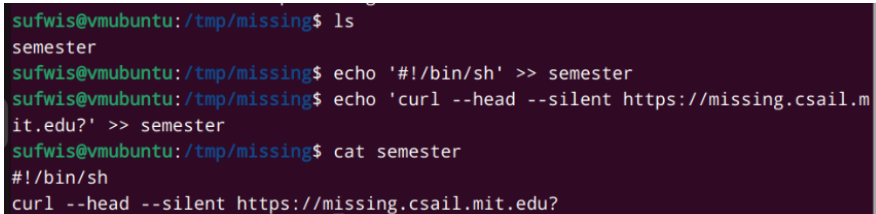
```
sufwis@vmubuntu:/$ touch /tmp/missing/semester
sufwis@vmubuntu:/$ ls /tmp/missing/semester
/tmp/missing/semester
sufwis@vmubuntu:/$
```

图 3: 如图, 创建成功

2.1.4 将以下内容一行一行地写入 semester 文件:

```
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu?
```

```
1 use vim or echo
2 use '' to include content if you choise echo
3 Because for bash, the content within single quotes will not be
   processed
```



```
sufwis@vmubuntu:/tmp/missing$ ls
semester
sufwis@vmubuntu:/tmp/missing$ echo '#!/bin/sh' >> semester
sufwis@vmubuntu:/tmp/missing$ echo 'curl --head --silent https://missing.csail.m
it.edu?' >> semester
sufwis@vmubuntu:/tmp/missing$ cat semester
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu?
```

图 4: 如图, 已经写入

2.1.5 尝试执行这个文件。例如, 将该脚本的路径 (./semester) 输入到您的 shell 中并回车?

```
1 first, we could find the file does not have execution permission
2 so, use chmod to add -x permission
3 finally, execute it.
```

```

sufwis@vmubuntu:/tmp/missing$ ll
总计 12
drwxrwxr-x  2 sufwis sufwis 4096  9月  5 11:57 ./
drwxrwxrwt 19 root   root   4096  9月  5 11:50 ../
-rw-rw-r--  1 sufwis sufwis  62  9月  5 11:55 semester
sufwis@vmubuntu:/tmp/missing$ chmod +111 semester
sufwis@vmubuntu:/tmp/missing$ ll
总计 12
drwxrwxr-x  2 sufwis sufwis 4096  9月  5 11:57 ./
drwxrwxrwt 19 root   root   4096  9月  5 11:50 ../
-rwxrwxr-x  1 sufwis sufwis  62  9月  5 11:55 semester*
sufwis@vmubuntu:/tmp/missing$ ./semester
HTTP/2 200
server: GitHub.com
content-type: text/html; charset=utf-8

```

图 5: 结果如图, 脚本执行后输出的部分内容被省略

2.1.6 查看 chmod 的手册(例如, 使用 man chmod 命令)?

```

1  man chmod
2  or
3  chmod --help

```

```

CHMOD(1)                                User Commands                                CHMOD(1)

NAME
    chmod - change file mode bits

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

```

图 6: 使用man chmod的结果

尽管 *man* 或者 *-help* 都能给出对应命令的解释, 给出对应的参数, 或者例子。但是如今, 使用 *ai* 插件询问也是推荐的。

2.1.7 使用 chmod 命令改变权限, 使 ./semester 能够成功执行, 不要使用 sh semester 来执行该程序。您的 shell 是如何知晓这个文件需要使用 sh 来解析呢?

首先, 如果要执行脚本, 一般有两种方式:

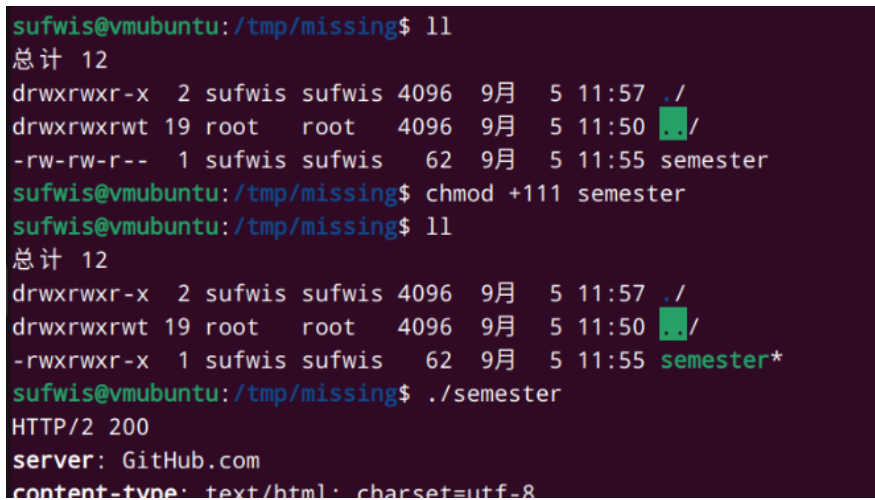
1. 解释器+脚本文件

2. ./脚本文件

第一种只会使用指定的解释器来执行脚本，且不需要脚本有可执行权限，第二种则会根据shebang执行，需求可执行权限。

因此，问题就明了了，先使用chmod为脚本提供可执行权限，然后./semester执行。

```
1  chmod +111 semester
2  ./semester
```

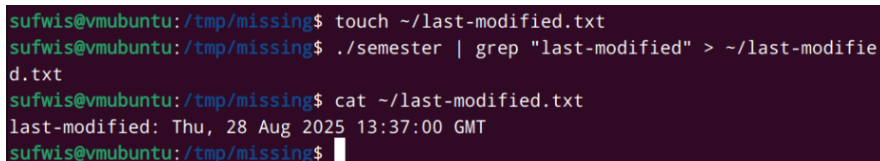


```
sufwis@vmubuntu:/tmp/missing$ ll
总计 12
drwxrwxr-x  2 sufwis sufwis 4096  9月  5 11:57 ./
drwxrwxrwt 19 root   root   4096  9月  5 11:50 ../
-rw-rw-r--  1 sufwis sufwis  62  9月  5 11:55 semester
sufwis@vmubuntu:/tmp/missing$ chmod +111 semester
sufwis@vmubuntu:/tmp/missing$ ll
总计 12
drwxrwxr-x  2 sufwis sufwis 4096  9月  5 11:57 ./
drwxrwxrwt 19 root   root   4096  9月  5 11:50 ../
-rwxrwxr-x  1 sufwis sufwis  62  9月  5 11:55 semester*
sufwis@vmubuntu:/tmp/missing$ ./semester
HTTP/2 200
server: GitHub.com
content-type: text/html; charset=utf-8
```

图 7: 已经执行过该脚本

2.1.8 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  touch ~/last-modified.txt # to make the file
2  ./semester | grep "last-modified" > ~/last-modified.txt
3  # execute semester, use grep to get lines that include "last-
    modified", re-direct
```



```
sufwis@vmubuntu:/tmp/missing$ touch ~/last-modified.txt
sufwis@vmubuntu:/tmp/missing$ ./semester | grep "last-modified" > ~/last-modified.txt
sufwis@vmubuntu:/tmp/missing$ cat ~/last-modified.txt
last-modified: Thu, 28 Aug 2025 13:37:00 GMT
sufwis@vmubuntu:/tmp/missing$
```

图 8: 如图

2.1.9 写一段命令来从 /sys 中获取笔记本的电量信息？

结论：

在物理 Linux 机器上，电量信息通常通过 sysfs 或 proc 文件系统提供，主要路径是：

`/sys/class/power_supply/`。

在这个目录下，你通常会看到像 BAT0（电池）或 AC（电源适配器）这样的文件夹。

在虚拟机中：

当你进入这个目录时，可能会有两种情况：目录为空：虚拟机没有模拟电池设备，因此没有相关信息。

有一个 AC 设备：虚拟机通常会模拟一个始终连接电源适配器的状态，让你感觉它一直在通电运行。

```
1 cd /sys/class/power_supply/ACAD
2 cat online
```

```
sufwis@vmubuntu:/sys/class/power_supply/ACAD$ ls
device extensions hwmon0 online power subsystem type uevent wakeup45
sufwis@vmubuntu:/sys/class/power_supply/ACAD$ cat online
1
sufwis@vmubuntu:/sys/class/power_supply/ACAD$
```

图 9: 图中输出1，表明电源联通

2.1.10 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（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 ls -a
2 ls -h
3 ls -t
4 ls --color=auto
```

```
Using color to distinguish file types is disabled both by default and
with --color=never. With --color=auto, ls emits color codes only when
standard output is connected to a terminal. The LS_COLORS environment
variable can change the settings. Use the dircolors(1) command to set
it.
```

图 10: 这是对-color参数的说明


```
sufwis@vmubuntu:/sys/class/power_supply/ACAD$ ls -laht --color=auto
总计 0
drwxr-xr-x 2 root root 0 9月 5 15:23 extensions
drwxr-xr-x 2 root root 0 9月 5 15:23 wakeup45
drwxr-xr-x 3 root root 0 9月 5 15:23 hwmon0
-r--r--r-- 1 root root 4.0K 9月 5 15:23 type
-r--r--r-- 1 root root 4.0K 9月 5 15:23 online
drwxr-xr-x 2 root root 0 9月 5 15:23 power
-rw-r--r-- 1 root root 4.0K 9月 5 15:23 uevent
drwxr-xr-x 3 root root 0 9月 5 15:23 ..
drwxr-xr-x 6 root root 0 9月 5 15:23 .
lrwxrwxrwx 1 root root 0 9月 5 08:29 device -> ../../../../ACPI0003:00
lrwxrwxrwx 1 root root 0 9月 5 08:28 subsystem -> ../../../../../../class/powe
supply
```

图 11: ls综合结果

2.2 bash脚本

2.2.1 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）加载函数涉及bash脚本语法和source加载脚本。source能够更改当前shell环境，例如修改变量、函数等。

```
1  #!/usr/bin/bash
2  marco_log_path="$HOME/logs/marco.log"
3  marco(){
4      echo "$(pwd)" > $marco_log_path
5      echo "pwd saved; use polo to back"
6  }
7  polo(){
8      cd "$(cat "$marco_log_path")"
9  }
```

```
sufwis@vmubuntu:~/learn/bashlearn/bashfiles$ source marco.sh
sufwis@vmubuntu:~/learn/bashlearn/bashfiles$ marco
pwd has saved
use polo to cd here
sufwis@vmubuntu:~/learn/bashlearn/bashfiles$ cd /
sufwis@vmubuntu:/$ polo
sufwis@vmubuntu:~/learn/bashlearn/bashfiles$
```

图 12: 演示，代码略有不同

2.2.2 编写一段 bash 脚本，完成以下任务？

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。

```
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"
```

下面是一种解决方法：

```
1  #!/usr/bin/env bash
2
3  if [[ -e sout.log ]]; then
4      rm sout.log -f
5  fi
6  touch sout.log
7  if [[ -e serr.log ]]; then
8      rm serr.log -f
9  fi
10 touch serr.log
11
12 echo "start~"
13 declare -i jud=0
14 declare -i tar=1
15 declare -i cnt=0
16 while [[ $jud -ne $tar ]];do
17     ./test.sh >> ./sout.log 2>> ./serr.log
18     jud=$?
19     cnt=$(( $cnt + 1 ))
20 done
21
```

```
22 echo "counts: $cnt"
```

```
sufwis@vmubuntu:~/learn/bashlearn/bashDebug$ ls
eq32.sh  test.sh
sufwis@vmubuntu:~/learn/bashlearn/bashDebug$ ./eq32.sh
start~
counts: 259
sufwis@vmubuntu:~/learn/bashlearn/bashDebug$ ls
eq32.sh  serr.log  sout.log  test.sh
sufwis@vmubuntu:~/learn/bashlearn/bashDebug$ cat serr.log
The error was using magic numbers
sufwis@vmubuntu:~/learn/bashlearn/bashDebug$ cat sout.log
Everything went according to plan
Everything went according to plan
Everything went according to plan
```

图 13: 结果如图

2.2.3 我们已经知道，命令行可以从参数或标准输入接受输入。在用管道连接命令时，我们将标准输出和标准输入连接起来。任务：

这里我们可以使用 `xargs` 命令，它可以使用标准输入中的内容作为参数。例如 `ls — xargs rm` 会删除当前目录中的所有文件。

您的任务是编写一个命令，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。

`-d` 指定 `xargs` 的分隔符。

```
1 find . -name "*.html" -print0 | xargs -d '\0' tar cvfz all.tar.gz
```

使用 `-print0` 以 `null` 字符作为输出的分隔符，`-d` 指定 `null` 字符，也可使用 `-0`。倘若使用默认的空白分隔符，含有空格的文件两侧分别被当两个文件处理。

```

├── dir1
│   └── h1.html
├── dir2
│   ├── h2.html
│   └── h3.html
├── dir3
│   ├── h1.html
│   └── h4.html
├── h1.html
└── have space.html

4 directories, 7 files
sufwis@vmubuntu:~/learn/bashlearn/htmlZip$ find . -name "*.html" -print0 | xargs
-d '\0' tar cvfz all.tar.gz
./dir3/h4.html
./dir3/h1.html
./h1.html
./dir1/h1.html
./dir2/h2.html
./dir2/h3.html
./have space.html
sufwis@vmubuntu:~/learn/bashlearn/htmlZip$ ls
all.tar.gz  dir1  dir2  dir3  h1.html  'have space.html'

```

图 14: 结果如图

2.2.4 编写一个命令或脚本递归的查找文件夹中最近修改的文件。更通用的做法，你可以按照最近的修改时间列出文件吗？

```

1 find . -mtime 0 -type f -print0 | xargs -0 ls -lt | head -5
2 # man find to check -mtime -type -print0 or man head

```

```

sufwis@vmubuntu:~/learn/bashlearn$ find . -mtime 0 -type f -print0 | xargs -0 ls
-lt | head -5
-rw-rw-r-- 1 sufwis sufwis 211  9月  5 17:42 ./htmlZip/all.tar.gz
-rw-rw-r-- 1 sufwis sufwis  0  9月  5 17:37 ./htmlZip/have space.html
-rw-rw-r-- 1 sufwis sufwis  0  9月  5 17:35 ./htmlZip/dir3/h1.html
-rw-rw-r-- 1 sufwis sufwis  0  9月  5 17:35 ./htmlZip/dir3/h4.html
-rw-rw-r-- 1 sufwis sufwis  0  9月  5 17:34 ./htmlZip/dir2/h3.html

```

图 15: 结果如图，查找一天内按照修改顺序排序的前5个文件

2.2.5 编写一个脚本，处理用户信息，演示字符串变量、数组和环境变量的使用？

```
sufwis@vmubuntu:~/learn/bashlearn/bashfiles$ ./var.sh
user: sufwis
home: /home/sufwis
pwd: /home/sufwis/learn/bashlearn/bashfiles
custom_user: a
hobbies:
~: read
~: write
~: eat
c: C
b: B
a: A
```

图 16: 结果如图

2.3 vim基础操作

2.3.1 完成 vimtutor?

```
1 vimtutor
```

```
=====
=   欢    迎    阅    读    《 V I M 教 程 》    —    版 本  1.7    =
=====

Vim 是一个具有很多命令的功能非常强大的编辑器。限于篇幅，在本教程当中
就不详细介绍了。本教程的设计目标是讲述一些必要的基本命令，而掌握好这
些命令，您就能够很容易地将 Vim 当作一个通用编辑器来使用了。

完成本教程的内容大约需要 25-30分钟，取决于您训练的时间。

注意：
每一节的命令操作将会更改本文。推荐您复制本文的一个副本，然后在副本上
进行训练(如果您是通过"vimtutor"来启动教程的，那么本文就已经是副本了)。

切记一点：本教程的设计思路是在使用中进行学习的。也就是说，您需要通过
执行命令来学习它们本身的正确用法。如果您只是阅读而不操作，那么您可能
会很快遗忘这些命令的！
```

图 17: 你将看到这样的画面，依照指示行动

2.3.2 下载我们的 vimrc，然后把它保存到 ~/.vimrc。通读这个注释详细的文件（用 Vim!），然后观察 Vim 在这个新的设置下看起来和使用起来有哪些细微的区别？

这个vimrc文件已经放在本目录下。

将之重命名为.vimrc并放在加目录下，之后使用vim打开，你将看到：

```

12
11 " Unbind some useless/annoying default key bindings.
10 nmap Q <Nop> " 'Q' in normal mode enters Ex mode. You almost never want this
9
8 " Disable audible bell because it's annoying.
7 set noerrorbells visualbell t_vb=
6
5 " Enable mouse support. You should avoid relying on this too much, but it ca
n
4 " sometimes be convenient.
3 set mouse+=a
2
1 " Try to prevent bad habits like using the arrow keys for movement. This is
57 " not the only possible bad habit. For example, holding down the h/j/k/l key
s
@

```

图 18: 参阅注释，尝试修改

例如 `set mouse+=a`，`a` 代表 "all"，意味着在所有模式（普通模式、插入模式、可视模式等）中都启用鼠标支持没有这个设置时，Vim 通常只在普通模式和可视模式中支持部分鼠标功能。

2.3.3 下载vim-plug?

克隆vim-plug仓库，将plug.vim移动到 `./vim/autoload/`下。之后修改 `./vimrc`。

在 `call plug#begin/end`之间，使用 `Plug 'url'`提供插件源，例如github、gitee仓库。之后 `source ./vimrc`应用这个文件的更改，执行 `:PlugInstall`下载，或者 `PlugUpdate`更新。

如果想要删除插件，只需要删除 `Plug 'url'`对应的那一行，然后 `:PlugClean`即可。

```

1      call plug#begin()
2      Plug 'preservim/NERDTree'
3      Plug 'wikitopian/hardmode'
4      .....
5      call plug#end()

```

```

1 Finished. 0 error(s).
2 [====]
3
4 - vim-easymotion: OK
5 - vim-airline: OK
6 - vim-commentary: OK
7 - nerdtree: OK
8 - auto-pairs: OK
9 call plug#begin('~/.vim/plugged')
10 " tree docs
11 Plug 'https://gitee.com/mirrors/nerdtree.git'
12 " beautiful status
13 Plug 'https://gitee.com/bingxuechanyaxvimer/vim-airline.git'
14 " auto pair ,like()
15 Plug 'https://gitee.com/mutiantian/auto-pairs.git'
16 " quick annotation
17 Plug 'https://gitee.com/yuet_tee/vim-commentary.git'
18 " vim-easymotion
19 Plug 'https://gitee.com/anchaos/vim-easymotion.git'
20
21
22
23 call plug#end()
24

```

图 19: 如图，安装插件后可以通过:PlugStatus查看插件状态

2.4 数据整理/正则表达式

2.4.1 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 # regular expr
2 ^([a]*a){3}.*(?<'s)$
3 # if use grep
4 cat /usr/share/dict/words | tr "[:upper:]" "[:lower:]" | grep -E
    "^([a]*a){3}.*$" | grep -v "'s$" | wc -l

```

```

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

```

图 20: grep不支持零宽断言

2.4.2 这些单词中，出现频率前三的末尾两个字母是什么?

```

1 cat /usr/share/dict/words | tr "[:upper:]" "[:lower:]" | grep -E
    "^([a]*a){3}.*$" | grep -v "'s$" |
2 sed -E "s/.*([a-z]{2})$/\1/" | sort | uniq -c | sort | tail -n5

```

```
sufwis@vmubuntu:/usr/share/dict$ cat ./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
```

图 21: sed用于将单词简化为末尾两个字母，然后排序统计

通过tail -n10 查看，ah频次同样是8，所以仅仅使用tail -n3是不够的。

3 实验感悟

shell/bash是计算机的一种文字接口，熟练使用能够缩短操作时间，不必使用鼠标，而是使用快捷的键盘。如果仅仅是为使用，学习shell并不复杂。只需要浏览一些介绍shell操作的博客即可。

然而事实上并非如此简单，如同vim一样，它们提供很高的自由度，这需要使用者付出时间学习。倘若想要更加顺手，或者自定义，更需要学习编写vim/shell本身的语言，例如使用vim script来编写插件。

vim是一个熟手觉得好用，新手认为落后的编辑器。如果习惯于各种预先提供好的服务，那么选用现代IDE是不错的选择；如果你认为IDE本身阻碍你的思考，让你的操作跟不上你的思维，那么vim更加便捷。

重要的是选择合适的，而非大众眼中最好的。因为合适的才是最好的。

4 个人github/gitee账号

[个人github账号](#)

[对应仓库](#)

5 参考资料

[计算机教育中缺失的一课](#)