

# 《系统开发工具基础》实验报告



题目: Shell 工具和脚本、vim、数据整理学习报告

学号: 23020007139

姓名: 闫枳冰

专业: 计算机科学与技术

日期: 2024 年 9 月 2 日

# 目录

<b>1 实验目的</b>	<b>1</b>
<b>2 实验内容及步骤</b>	<b>1</b>
2.1 实验主要内容	1
2.2 实验具体步骤	1
2.2.1 shell 工具和脚本	1
2.2.2 编辑器 vim	3
2.2.3 数据整理	4
2.2.4 实验记录	4
<b>3 实例</b>	<b>4</b>
3.1 实例一: 在 /tmp 下新建一个名为 missing 的文件夹。	4
3.2 实例二: 用 man 查看程序 touch 的使用手册	4
3.3 实例三: 用 touch 在 missing 文件夹中新建一个叫 semester 的文件。	5
3.4 实例四: 将以下内容一行一行地写入 semester 文件: 第一行 #!/bin/sh 第二行 curl -head -silent https://missing.csail.mit.edu	5
3.5 实例五: 将该脚本的路径 (./semester) 输入到您的 shell 中并回车。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。	5
3.6 实例六: 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。	6
3.7 实例七: 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	6
3.8 实例八: 写一段命令来从 /sys 中获取笔记本的电量信息, 或者台式机 CPU 的温度	6
3.9 实例九: 在 missing 文件夹中创建 a.txt, 用 mv 改名为 b.txt, 在用 cp 将 b.txt 复制到 tmp 文件夹下	7
3.10 实例十: 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	7
3.11 实例十一: 编写两个 bash 函数 marco 和 polo 执行下面的操作。	7
3.12 实例十二: 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。	7
3.13 实例十三: 编写一个 Shell 脚本, 列出当前目录下所有文件的大小	9
3.14 实例十四: 安装和配置一个插件: ctrlp.vim.	9
3.15 实例十五: 在 Vim 中, 将第 5 行的内容复制到最后一行之后。	9
3.16 实例十六: 进一步自定义你的 ~/.vimrc 和安装更多插件	9
3.17 实例十七: 自定义 CtrlP: 添加 configuration 到你的 ~/.vimrc 来用按 Ctrl-P 打开 CtrlP	10
3.18 实例十八: 输出文本带有电话的行	10
3.19 实例十九: 使用 grep 来搜索文本	10
3.20 实例二十: 搜索 a 开头的行	10
<b>4 实验结果</b>	<b>11</b>
<b>5 解题感悟</b>	<b>11</b>

# 1 实验目的

## 2 实验内容及步骤

### 2.1 实验主要内容

1. 了解学习 shell 工具，掌握 Shell 脚本的基本语法和编写方法。2. 了解学习 Vim 编辑器的使用。3. 认识数据整理的重要性，学习数据整理。

### 2.2 实验具体步骤

#### 2.2.1 shell 工具和脚本

1. 在 windows 下载 wsl。

在 powershell 中输入“wsl -install”，安装完成后打开 Ubuntu 创建用户名和密码。

2. 使用 shell。

(1) 打开 Ubuntu 后，输入 date，可以查看当前时间。如图1

```
yzb0202@yzb0202:~$ date
Sun Sep  1 20:32:30 CST 2024
```

图 1: date

(2) 让 shell 执行 echo 命令，了解到 shell 执行不是关键字的命令是如何执行的。即咨询环境变量 \$PATH 来执行命令。如图2。

```
yzb0202@yzb0202:~$ echo hello
hello
yzb0202@yzb0202:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/ssl/lib:/mnt/c/Windows/System32:/mnt/c/Windows/System32/Wbem:/mnt/c/Windows/System32/WindowsPowerShell/v1.0:/mnt/c/Windows/System32/OpenSSH:/mnt/d/Windows Kits/10/Windows Performance Toolkit:/mnt/d/download/Git/cmd:/mnt/c/Users/22746/AppData/Local/Microsoft/WindowsApps/snap/bin
yzb0202@yzb0202:~$ which echo
/usr/bin/echo
yzb0202@yzb0202:~$ /bin/echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/ssl/lib:/mnt/c/Windows/System32:/mnt/c/Windows/System32/Wbem:/mnt/c/Windows/System32/WindowsPowerShell/v1.0:/mnt/c/Windows/System32/OpenSSH:/mnt/d/Windows Kits/10/Windows Performance Toolkit:/mnt/d/download/Git/cmd:/mnt/c/Users/22746/AppData/Local/Microsoft/WindowsApps/snap/bin
yzb0202@yzb0202:~$
```

图 2: echo

3. 在 shell 中导航

在 linux 中路径被/分割，路径/代表根目录，当前工作目录可以使用 pwd 命令获取，切换目录用 cd 命令，. 表示当前目录，.. 表示上一级目录。如图3

```
yzb0202@yzb0202:~$ pwd
/home/yzb0202
yzb0202@yzb0202:~$ cd /home
yzb0202@yzb0202:/home$ pwd
/home
yzb0202@yzb0202:/home$ cd ..
yzb0202@yzb0202:/$ pwd
/
```

图 3: cd

ls 命令可以查看打印目录下的文件，ls -l 可以更详细的列出信息。如图4. 打印出的信息中 d 表示是一个目录，后每三个字母为一组，第一个字母代表文件所有者，第二个代表用户组，第三个代表其他所有人。r 代表只读，w 代表可修改，x 代表可执行。

```

yzb0202@yzb0202:/$ ls
bin    dev    home   lib    lib64  lost+found  mnt    proc  run    snap  sys  usr
boot   etc    init   lib32  libx32  media      opt    root  sbin  srv   tmp  var
yzb0202@yzb0202:/$ ls -l
total 2144
lrwxrwxrwx 1 root root 7 Apr 23 2020 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Apr 23 2020 boot
drwxr-xr-x 11 root root 3060 Sep 1 20:32 dev
drwxr-xr-x 92 root root 4096 Sep 1 20:32 etc
drwxr-xr-x 3 root root 4096 Aug 30 13:32 home
-rwxrwxrwx 1 root root 2127224 Apr 26 02:17 init
lrwxrwxrwx 1 root root 7 Apr 23 2020 lib -> usr/lib
lrwxrwxrwx 1 root root 9 Apr 23 2020 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 Apr 23 2020 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Apr 23 2020 libx32 -> usr/libx32
drwx----- 2 root root 16384 Aug 30 13:31 lost+found
drwxr-xr-x 2 root root 4096 Apr 23 2020 media
drwxr-xr-x 6 root root 4096 Aug 30 13:32 mnt
drwxr-xr-x 2 root root 4096 Apr 23 2020 opt
dr-xr-xr-x 232 root root 0 Sep 1 20:32 proc
drwx----- 2 root root 4096 Apr 23 2020 root
drwxr-xr-x 7 root root 140 Sep 1 20:32 run
lrwxrwxrwx 1 root root 8 Apr 23 2020 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Apr 10 2020 snap
drwxr-xr-x 2 root root 4096 Apr 23 2020 srv
dr-xr-xr-x 11 root root 0 Sep 1 20:32 sys
drwxrwxrwt 3 root root 4096 Sep 1 20:32 tmp
drwxr-xr-x 14 root root 4096 Apr 23 2020 usr
drwxr-xr-x 13 root root 4096 Apr 23 2020 var

```

图 4: ls

mv 命令用于重命名或者移动文档，cp 用于拷贝文件，mkdir 用于新建文件。如图5。在 missing 文件夹中创建 a.txt, 用 mv 改名为 b.txt, 在用 cp 将 b.txt 复制到 tmp 文件夹下，mkdir 创建 b 文件夹。

```

yzb0202@yzb0202:/tmp/missing$ echo abc > a.txt
yzb0202@yzb0202:/tmp/missing$ ls
a.txt
yzb0202@yzb0202:/tmp/missing$ mv a.txt b.txt
yzb0202@yzb0202:/tmp/missing$ ls
b.txt
yzb0202@yzb0202:/tmp/missing$ cp b.txt /tmp
yzb0202@yzb0202:/tmp/missing$ cd ..
yzb0202@yzb0202:/tmp$ ls
b.txt  missing
yzb0202@yzb0202:/tmp$ cd missing
yzb0202@yzb0202:/tmp/missing$ mkdir a
yzb0202@yzb0202:/tmp/missing$ ls
a  b.txt
yzb0202@yzb0202:/tmp/missing$

```

图 5: mv

4. 在程序间创建连接。

shell 中有输入流和输出流，但可以重定向这些流，如 < file 和 > file,» 用于追加文件。如图6。

5. 使用管道将命令组合使用。”中定义的字符串变量不会被转义，””定义的字符串中变量值会被替

```

yzb0202@yzb0202:/tmp/missing$ echo hello > hello.txt
yzb0202@yzb0202:/tmp/missing$ cat < hello.txt
hello
yzb0202@yzb0202:/tmp/missing$ cat <hello.txt > hello2.txt
yzb0202@yzb0202:/tmp/missing$ cat < hello2.txt
hello

```

图 6: stream

换。\$\$ 为与操作符，|| 为或操作符，一行中多个命令用；分割。用 \$(命令) 可以进行命令替换，如图7。

6. 查看命令如何使用。

输入 man+ 命令或者为对应的命令行添加 -h 或 -help 标记来查看命令如何使用。

7. 查找文件。

使用 find 命令来递归搜索符合条件的文件。如图8、9、10。

8. 查找代码。可以使用 grep 或者 rg、ag、ack 等命令。如图11 9. 查找 shell 命令。可以用 grep 进行模式搜索，也可以使用 Ctrl+R 对命令历史记录进行回溯搜索。如图12。

```

yzb0202@yzb0202:/tmp/missing$ foo=bar
yzb0202@yzb0202:/tmp/missing$ echo "$foo"
bar
yzb0202@yzb0202:/tmp/missing$ echo '$foo'
$foo
yzb0202@yzb0202:/tmp/missing$ false || echo "Oops, fail"
Oops, fail
yzb0202@yzb0202:/tmp/missing$ true && echo "Things went well"
Things went well
yzb0202@yzb0202:/tmp/missing$ echo "Starting program at $(date)"
Starting program at Sun Sep  1 23:12:00 CST 2024
yzb0202@yzb0202:/tmp/missing$

```

图 7: 组合

```

yzb0202@yzb0202:~$ echo "123" > 2.txt
yzb0202@yzb0202:~$ echo "123" > 3.txt
yzb0202@yzb0202:~$ echo "123" > 4.txt
yzb0202@yzb0202:~$ echo "123" > 5.txt
yzb0202@yzb0202:~$ find . -name words -type d
./words
yzb0202@yzb0202:~$ find . -path '*/words/*.txt' -type f
./words/1.txt
./words/5.txt
./words/3.txt
./words/2.txt
./words/4.txt
yzb0202@yzb0202:~$ find . -mtime -1
.
./words

```

图 8: find

```

yzb0202@yzb0202:~$ find . -mtime -1
.
./tmp
./tmp/missing
./tmp/missing/semester
./tmp/missing/cat
./profile
./motd_shown
./landscape
./landscape/sysinfo.log
./last-modified.txt
./bash_history
./bashrc
./bash_logout

```

图 9: find

```

yzb0202@yzb0202:~$ find . -name '*.txt' -exec rm {} \;
yzb0202@yzb0202:~$ ls
tmp  words
yzb0202@yzb0202:~$ cd words/
yzb0202@yzb0202:~/words$ ls
yzb0202@yzb0202:~/words$ |

```

图 10: find

## 2.2.2 编辑器 vim

### 1. 编辑模式。

vim 有正常模式、插入模式、替换模式、可视化模式、命令模式五种模式。左下角会显示当前模式，按 <ESC> 返回正常模式，在正常模式中键入 i 进入插入模式，r 进入替换模式，v 进入可视模式，: 进去命令模式。

### 2. 基本操作。

键入 i 后进入插入模式可正常编辑，直到键入 <ESC> 返回正常模式。一个 Vim 会话包含一系列标签页，每个标签页包含一系列窗口。每个窗口显示一个缓存。进入命令行模式后输入 q 表示退

```

yzb0202@yzb0202:/tmp/missing$ rg -u --files-without-match "
#!/
hello2.txt
b.txt
hello.txt
yzb0202@yzb0202:/tmp/missing$ echo "foo" >> hello.txt
yzb0202@yzb0202:/tmp/missing$ rg foo -A 5
hello.txt
2:foo
yzb0202@yzb0202:/tmp/missing$ rg --stats PATTERN

0 matches
0 matched lines
0 files contained matches
3 files searched
0 bytes printed
20 bytes searched
0.000023 seconds spent searching
0.004070 seconds

```

图 11: code

```

yzb0202@yzb0202:~$ history | grep find
42 find . -mtime -1
43 find . -path '*/1/*.c' -type f
44 find . -name pic -type d
45 find . -name learn-git -type d
46 find . -name shell -type d
47 find . -size +500k -size -10M -name '*.tar.g
50 find . -name words -type d
51 find . -path '*/words/*.txt' -type f
52 find . -mtime -1
53 find . -name '*.txt' -exec rm {} \;
177 history | grep find
yzb0202@yzb0202:~$

```

图 12: history

出，w 表示保存，wq 表示退出并保存，e{文件名} 表示要打开的文件，ls 显示打开的缓存。

### 3. 命令使用。

hjkl 进行基本移动，w 表示下一个词，b 表示词初，e 表示词尾，0 表示行初等；v 表示可视化，V 表示可视化行，Ctrl+v 表示可视化块；O / o 在之上/之下插入行，d 移动命令 删除移动命令等；可以用一个计数来结合“名词”和“动词”，这会执行指定操作若干次；你可以用修饰语改变“名词”的意义。修饰语有 i，表示“内部”或者“在内”，和 a，表示“周围”。

### 4. 其他见实例。

## 2.2.3 数据整理

学习数据整理的内容，学习正则表达式，具体见实例。

## 2.2.4 实验记录

链接：[练习记录](#)

# 3 实例

## 3.1 实例一：在 /tmp 下新建一个名为 missing 的文件夹。

使用 mkdir+ 文件名的命令。如图13。

## 3.2 实例二：用 man 查看程序 touch 的使用手册

使用命令 man +touch。如图14

```
yzb0202@yzb0202:/tmp$ mkdir missing
yzb0202@yzb0202:/tmp$ ls
b.txt  missing
```

图 13: mkdir

```
yzb0202@yzb0202:/tmp$ touch --help
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
given.
A FILE argument string of - is handled specially and causes touch to
write to standard output.
Mandatory arguments to long options are mandatory for short options too.
-a      change only the access time
-c, --no-create
        do not create any files
-d, --date=STRING
        parse STRING and use it instead of current time
-f      (ignored)
-h, --no-dereference
        do not dereference symbolic links
Manual page touch(1) line 1 (press h for help or q to quit)
```

图 14: touch

### 3.3 实例三：用 touch 在 missing 文件夹中新建一个叫 semester 的文件。

如图15，使用命令 touch+ 文件名

```
yzb0202@yzb0202:/tmp$ cd missing
yzb0202@yzb0202:/tmp/missing$ touch semester
yzb0202@yzb0202:/tmp/missing$ echo '#!/bin/sh' > semester
yzb0202@yzb0202:/tmp/missing$ echo 'curl --head --silent https://missing.csail.mit.edu' >> semester
yzb0202@yzb0202:/tmp/missing$ cat semester
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu
```

图 15: touch

### 3.4 实例四：将以下内容一行一行地写入 semester 文件：第一行 #!/bin/sh 第二行 curl --head --silent https://missing.csail.mit.edu

用命令 echo+ 内容 + » 或者 > + 文件名，> 是重新写入，» 是追加。内容用” 引用不转移。

```
yzb0202@yzb0202:/tmp$ cd missing
yzb0202@yzb0202:/tmp/missing$ touch semester
yzb0202@yzb0202:/tmp/missing$ echo '#!/bin/sh' > semester
yzb0202@yzb0202:/tmp/missing$ echo 'curl --head --silent https://missing.csail.mit.edu' >> semester
yzb0202@yzb0202:/tmp/missing$ cat semester
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu
```

图 16: touch

### 3.5 实例五：将该脚本的路径 (./semester) 输入到您的 shell 中并回车。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。

如图17，不能执行的原因是没有执行权限。

```

yzb0202@yzb0202:~/tmp/missing$ ./semester
-bash: ./semester: Permission denied
yzb0202@yzb0202:~/tmp/missing$ ls
semester
yzb0202@yzb0202:~/tmp/missing$ ls -l
total 4
-rw-r--r-- 1 yzb0202 yzb0202 61 Aug 30 11:04 semester

```

图 17: semester

### 3.6 实例六: 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。

如图18。

```

yzb0202@yzb0202:~/tmp/missing$ cat semester
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu
yzb0202@yzb0202:~/tmp/missing$ ./semester
-bash: ./semester: Permission denied
yzb0202@yzb0202:~/tmp/missing$ ls
semester
yzb0202@yzb0202:~/tmp/missing$ ls -l
total 4
-rw-r--r-- 1 yzb0202 yzb0202 61 Aug 30 11:04 semester
yzb0202@yzb0202:~/tmp/missing$ man chmod
yzb0202@yzb0202:~/tmp/missing$ man chmod
yzb0202@yzb0202:~/tmp/missing$ man chmod
yzb0202@yzb0202:~/tmp/missing$ chmod 777 semester
yzb0202@yzb0202:~/tmp/missing$ ./semester
HTTP/2 200
server: GitHub.com
content-type: text/html; charset=utf-8
last-modified: Thu, 08 Aug 2024 20:16:01 GMT
access-control-allow-origin: *
etag: "66b52781-205d"
expires: Mon, 26 Aug 2024 19:04:44 GMT
cache-control: max-age=600
x-proxy-cache: MISS
x-github-request-id: 4A7C:81A35:72A77:769D8:66CCCF74
accept-ranges: bytes
age: 72
date: Fri, 30 Aug 2024 03:40:18 GMT
via: 1.1 varnish
x-served-by: cache-nrt-rjtf7700072-NRT

```

图 18: semester

### 3.7 实例七: 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

如图19。

```

yzb0202@yzb0202:~/tmp/missing$ ./semester | grep last-modified > ~/last-modified.txt
yzb0202@yzb0202:~/tmp/missing$ cat ~/last-modified.txt
last-modified: Thu, 08 Aug 2024 20:16:01 GMT
yzb0202@yzb0202:~/tmp/missing$

```

图 19: modified

### 3.8 实例八: 写一段命令来从 /sys 中获取笔记本的电量信息, 或者台式机 CPU 的温度

如图20。



```

yzb0202@yzb0202:~/tmp/missing$ cat /sys/class/power_supply/BAT1/capacity
23
yzb0202@yzb0202:~/tmp/missing$

```

图 20: power

### 3.9 实例九：在 missing 文件夹中创建 a.txt, 用 mv 改名为 b.txt, 在用 cp 将 b.txt 复制到 tmp 文件夹下

如图21所示，用 mv，cp 指令进行操作。

```

yzb0202@yzb0202:~/tmp/missing$ echo abc > a.txt
yzb0202@yzb0202:~/tmp/missing$ ls
a.txt
yzb0202@yzb0202:~/tmp/missing$ mv a.txt b.txt
yzb0202@yzb0202:~/tmp/missing$ ls
b.txt
yzb0202@yzb0202:~/tmp/missing$ cp b.txt /tmp
yzb0202@yzb0202:~/tmp/missing$ cd ..
yzb0202@yzb0202:/tmp$ ls
b.txt  missing
yzb0202@yzb0202:/tmp$ cd missing
yzb0202@yzb0202:~/tmp/missing$ mkdir a
yzb0202@yzb0202:~/tmp/missing$ ls
a  b.txt
yzb0202@yzb0202:~/tmp/missing$

```

图 21: mv

### 3.10 实例十：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

如图22

```

yzb0202@yzb0202:/$ ls -a
.  bin  dev  home  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr
.. boot  etc  init  lib32  libx32  media  opt  root  sbin  srv  tmp  var
yzb0202@yzb0202:/$ ls -h
bin  dev  home  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr
boot  etc  init  lib32  libx32  media  opt  root  sbin  srv  tmp  var
yzb0202@yzb0202:/$ ls --t
ls: option '--t' is ambiguous; possibilities: '--tabsize' '--time' '--time-style'
Try 'ls --help' for more information.
yzb0202@yzb0202:/$ ls -t
tmp  run  proc  home  lost+found  boot  root  media  srv  lib32  lib  bin
etc  dev  sys  mnt  init  var  usr  opt  libx32  lib64  sbin  snap
yzb0202@yzb0202:/$ ls --color=auto
bin  dev  home  lib  lib64  lost+found  mnt  proc  run  snap  sys  usr
boot  etc  init  lib32  libx32  media  opt  root  sbin  srv  tmp  var

```

图 22: l

### 3.11 实例十一：编写两个 bash 函数 marco 和 polo 执行下面的操作。

具体操作：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。如图23和24

### 3.12 实例十二：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。

如图25和26。



### 3.13 实例十三: 编写一个 Shell 脚本, 列出当前目录下所有文件的大小

先使用 `find` 命令查找当前目录下的所有文件再通过 `-exec` 选项对它们执行 `ls -lh` 命令。如图27。输入命令 `find . -maxdepth 1 -type f -exec ls -lh {} \;`; 其中 `.` 表示当前目录, `-maxdepth 1` 表示只搜索当前目录, `-type f` 表示只查找普通文件, `find` 命令都会执行 `-exec` 后面指定的命令。

```
yzb0202@yzb0202:~/tmp/missing$ echo "222222222222" > 2.txt
yzb0202@yzb0202:~/tmp/missing$ find . -maxdepth 1 -type f -exec ls -lh {} \;
-rw-r--r-- 1 yzb0202 yzb0202 210 Aug 31 17:55 ./debug.sh
-rw-r--r-- 1 yzb0202 yzb0202 124 Sep  2 14:52 ./1.txt
-rwxrwxrwx 1 yzb0202 yzb0202 61 Aug 30 13:46 ./semester
-rwxrwxrwx 1 yzb0202 yzb0202 212 Aug 31 17:54 ./tests.sh
-rw-r--r-- 1 yzb0202 yzb0202 1.4K Aug 31 17:55 ./out.log
-rw-r--r-- 1 yzb0202 yzb0202 13 Sep  2 15:12 ./2.txt
```

图 27: b

### 3.14 实例十四: 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`  
阅读这个插件的文档。
3. 尝试用 `CtrlP` 来在一个工程文件夹里定位一个文件, 打开 `Vim`, 然后用 `Vim` 命令控制行开始: `CtrlP`. 如图28和29。

```
yzb0202@yzb0202:~$ mkdir -p ~/.vim/pack/vendor/start
yzb0202@yzb0202:~$ cd ~/.vim/pack/vendor/start
yzb0202@yzb0202:~/.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 | 1.63 MiB/s, done.
Resolving deltas: 100% (1661/1661), done.
```

图 28: Caption

```
> 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 <mr>=< files <buf> <-> /home/yzb0202/.vim/pack/vendor/start
>>> _
```

图 29: Caption

### 3.15 实例十五: 在 Vim 中, 将第 5 行的内容复制到最后一行之后。

将光标移动到第 5 行, 按下 `yy` 复制当前行, 将光标移动到第 10 行, 按下 `p` 将之前复制的行粘贴到当前行的下一行。30

### 3.16 实例十六: 进一步自定义你的 `/.vimrc` 和安装更多插件

1. 安装 `vim-plug`。输入命令 `curl -fLo ~/.vim/autoload/plug.vim --create-dirs https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim`
2. 修改 `/.vimrc`。在其中输入如图代码31
3. 命令行中执行: `PlugInstall`。

```
yzb0202@yzb0202: ~/tmp/mi × + v
1号的电话是123456-7777, 2号的电话是333333-1234,
abc
电话
(123) 456-7890
(222)223-2344
(222) 223-2344
abcdefg
123 456-7890
~
~
```

图 30: vimp

```
yzb0202@yzb0202: ~/tmp/mi × + v
call plug#begin()
Plug 'preservim/NERDTree'
Plug 'wikitopian/hardmode'
call plug#end
()set runtimepath^=~/vim/pack/vendor/start/ctrlp.vim
~
~
~
```

图 31: vimrc

### 3.17 实例十七: 自定义 CtrlP: 添加 configuration 到你的 ~/.vimrc 来用按 Ctrl-P 打开 CtrlP

如图32

```
yzb0202@yzb0202: ~/.vim/pe × + v
let g:ctrlp_map = '<c-p>'
let g:ctrlp_cmd = 'CtrlP'
let g:ctrlp_working_path_mode = 'ra'
```

图 32: ctrlp

### 3.18 实例十八: 输出文本带有电话的行

如图33, 使用 awk '/正则表达式/ 动作' 文件名来利用正则表达式输出。

```
yzb0202@yzb0202:~/tmp$ cd missing
yzb0202@yzb0202:~/tmp/missing$ echo "1号的电话是123456-7777, 2号的电话是333333-1234, " >> 1.txt
yzb0202@yzb0202:~/tmp/missing$ awk '/电话/ {print}' 1.txt
1号的电话是123456-7777, 2号的电话是333333-1234,
yzb0202@yzb0202:~/tmp/missing$ echo "abc" >> 1.txt
yzb0202@yzb0202:~/tmp/missing$ echo "电话" >> 1.txt
yzb0202@yzb0202:~/tmp/missing$ awk '/电话/ {print}' 1.txt
1号的电话是123456-7777, 2号的电话是333333-1234,
电话
```

图 33: 电话

### 3.19 实例十九: 使用 grep 来搜索文本

搜索文本中符合“(222) 123-3333”格式。如图34, grep 默认不支持括号 () 的分组功能, 所以使用-E 选项, 需要将 d 替换为 [0-9]。

### 3.20 实例二十: 搜索 a 开头的行

输入 “grep ‘^a’ 文件名” 命令行。

```

yzb0202@yzb0202:~/tmp/missing$ grep '\\(\\d{3}\\)\\d{3}\\d{4}' 1.txt
yzb0202@yzb0202:~/tmp/missing$ grep -E '\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}' 1.txtgrep -E '\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}' 1.txt
txt
grep: 1.txt: No such file or directory
grep: '\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}': No such file or directory
1.txt:(123) 456-7890
yzb0202@yzb0202:~/tmp/missing$ grep -E '\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}' 1.txt
(123) 456-7890
yzb0202@yzb0202:~/tmp/missing$ echo (222)223-2344 >> 1.txt
-bash: syntax error near unexpected token `222'
yzb0202@yzb0202:~/tmp/missing$ echo '(222)223-2344' >> 1.txt
yzb0202@yzb0202:~/tmp/missing$ grep -E '\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}' 1.txt
(123) 456-7890
yzb0202@yzb0202:~/tmp/missing$ echo '(222) 223-2344' >> 1.txt
yzb0202@yzb0202:~/tmp/missing$ grep -E '\\([0-9]{3}\\) [0-9]{3}-[0-9]{4}' 1.txt
(123) 456-7890
(222) 223-2344

```

图 34: tp

```

yzb0202@yzb0202:~/tmp/missing$ cat 1.txt
1号的电话是123456-7777, 2号的电话是333333-1234,
abc
电话
(123) 456-7890
(222)223-2344
(222) 223-2344
yzb0202@yzb0202:~/tmp/missing$ echo abcdefg >> 1
yzb0202@yzb0202:~/tmp/missing$ grep '^a' 1.txt
abc
abcdefg
yzb0202@yzb0202:~/tmp/missing$

```

图 35: a

## 4 实验结果

1. 学习了解了 shell 脚本，从基本的如何常见文件夹到写入文件再到修改权限、函数调用等。
2. 学习了如何使用 vim 编辑器，认识到了 vim 编辑器的快捷方便。
3. 学习了如何利用正则表达式进行数据整理。
4. 实验报告及记录链接：<https://github.com/yyy0202/-remote-repo/tree/main/>

## 5 解题感悟

此次学习，让我深刻体会到 shell 脚本的强大，能够提升工作效率，尤其是在批量处理文件上。Vim 编辑器则让我领略到高效编辑的精髓，在掌握后，快捷键和命令模式能让我们编辑更快速。数据整理，通过掌握正则表达式，我学会了从繁杂的数据中提取有价值的信息。这我对数据处理有了更深的理解。我也意识到想要更加熟练地掌握还需要不断地练习。