

# 系统开发工具基础实验报告

23090031040 王璐鑫

2025 年 9 月 5 日

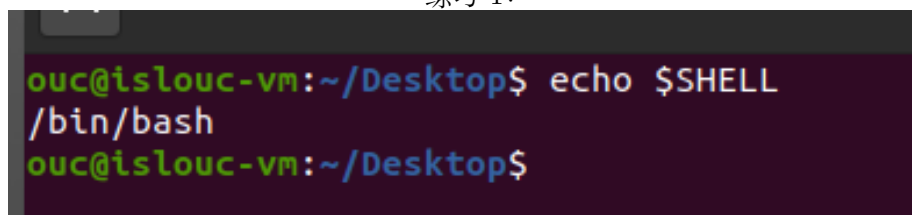
## 1 练习内容

1. SHEll 的安装与配置检查
2. 在 /tmp 下新建一个名为 missing 的文件夹
3. 用 man 查看程序 touch 的使用手册
4. 用 touch 在 missing 文件夹中新建一个叫 semester 的文件
5. 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`
6. 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。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。
7. 查看 `chmod` 的手册，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。
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
9. 写一段命令来从 `/sys` 中获取笔记本的电量信息
10. 阅读 `man ls`，然后使用 `ls` 命令进行如下操作：
  1. 所有文件（包括隐藏文件）
  2. 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`）
  3. 文件以最近修改顺序排序
  4. 以彩色文本显示输出结果

11. 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。
12. 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。
13. 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
14. 下载我们提供的 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。
15. vimtutor 第一讲
16. vimtutor 第二讲
17. vimtutor 第三讲
18. vimtutor 第四讲
19. 学习一下这篇简短的交互式正则表达式教程（1-8）
20. 学习一下这篇简短的交互式正则表达式教程（9-16）

## 2 结果展示

练习 1:



```
ouc@islouc-vm:~/Desktop$ echo $SHELL
/bin/bash
ouc@islouc-vm:~/Desktop$
```

### 练习 2:

```
ouc@islouc-vm:~/Desktop$ cd tmp
ouc@islouc-vm:~/Desktop/tmp$ mkdir missing
ouc@islouc-vm:~/Desktop/tmp$ ls
missing
ouc@islouc-vm:~/Desktop/tmp$
```

### 练习 3:

```
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

    -c, --no-create
            do not create any files

    -d, --date=STRING
            parse STRING and use it instead of current time

    -f      (ignored)
Manual page touch(1) line 8 (press h for help or q to quit)
```

### 练习 4:

```
ouc@islouc-vm:~/Desktop/tmp/missing$ touch semester
```

### 练习 5:

```
ouc@islouc-vm:~/Desktop/tmp/missing$ echo '#!/bin/sh' > semester
ouc@islouc-vm:~/Desktop/tmp/missing$ echo 'curl --head --silent https://missing
.csail.mit.edu' >> semester
ouc@islouc-vm:~/Desktop/tmp/missing$ cat semester
#!/bin/sh
curl --head --silent https://missing.csail.mit.edu
```

练习 6:

```
ouc@islouc-vm:~/Desktop/tmp/missing$ ./semester
bash: ./semester: Permission denied
Burp Suite Community Edition 1 9月 5 09:27 semester
ouc@islouc-vm:~/Desktop/tmp/missing$
```

练习 7:

```
ouc@islouc-vm:~/Desktop/tmp/missing$ man chmod
ouc@islouc-vm:~/Desktop/tmp/missing$ chmod +x semester
ouc@islouc-vm:~/Desktop/tmp/missing$ ls -l semester
-rwxrwxr-x 1 ouc ouc 61 9月 5 09:27 semester
ouc@islouc-vm:~/Desktop/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: Thu, 04 Sep 2025 15:52:05 GMT
cache-control: max-age=600
x-proxy-cache: MISS
x-github-request-id: 29FC:3C4A54:29DDD0:2C3AC4:68B9B343
accept-ranges: bytes
age: 0
date: Fri, 05 Sep 2025 01:37:13 GMT
via: 1.1 varnish
x-served-by: cache-tyo11975-TYO
x-cache: HIT
x-cache-hits: 0
x-timer: S1757036234.742381,VS0,VE182
vary: Accept-Encoding
x-fastly-request-id: 67425a4b458154eef98cd590aa51806870c05be6
content-length: 8194
```

练习 8:

```
ouc@islouc-vm:~/Desktop/tmp/missing$ ./semester | grep -i '^last-modified:' > ~/last-modified.txt
ouc@islouc-vm:~/Desktop/tmp/missing$
ouc@islouc-vm:~/Desktop/tmp/missing$ ls -l ~/last-modified.txt
-rw-rw-r-- 1 ouc ouc 46 9月 5 10:13 /home/ouc/last-modified.txt
ouc@islouc-vm:~/Desktop/tmp/missing$ cat ~/last-modified.txt
last-modified: Thu, 28 Aug 2025 13:37:00 GMT
ouc@islouc-vm:~/Desktop/tmp/missing$
```

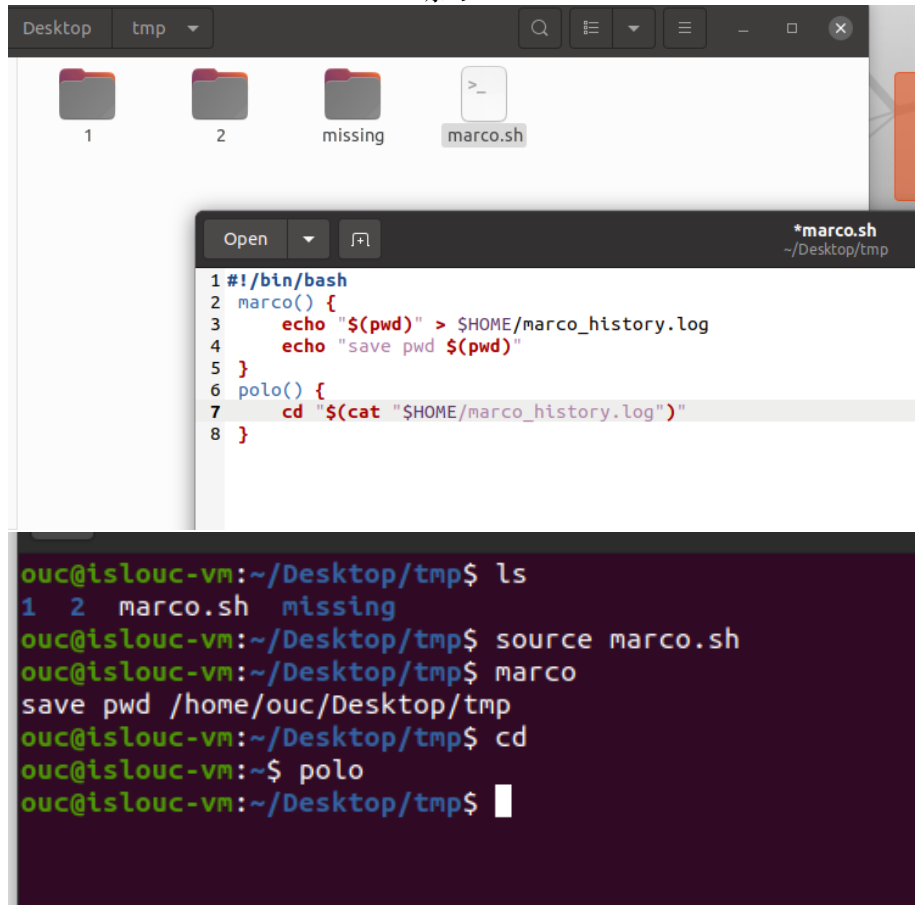
### 练习 9:

```
ouc@islouc-vm:~/Desktop/tmp$ ls /sys/class/power_supply/ACAD
device hwmmon0 online power subsystem type uevent wakeup41
ouc@islouc-vm:~/Desktop/tmp$ ls /sys/class/power_supply/ACAD/device
driver hid modalias path physical_node power power_supply status subsystem uevent uid
ouc@islouc-vm:~/Desktop/tmp$ cat /sys/class/power_supply/ACAD/device/status
15
```

### 练习 10:

```
ouc@islouc-vm:~/Desktop/tmp$ ls -a
. .. 1 2 missing
ouc@islouc-vm:~/Desktop/tmp$ ls -h
1 2 missing
ouc@islouc-vm:~/Desktop/tmp$ ls -l
total 12
drwxrwxr-x 2 ouc ouc 4096 9月 5 10:24 1
drwxrwxr-x 2 ouc ouc 4096 9月 5 10:24 2
drwxrwxr-x 2 ouc ouc 4096 9月 5 09:14 missing
ouc@islouc-vm:~/Desktop/tmp$ ls -t
2 1 missing
ouc@islouc-vm:~/Desktop/tmp$ ls --color
1 2 missing
ouc@islouc-vm:~/Desktop/tmp$ ls -laht --color=always
total 20K
drwxrwxr-x 5 ouc ouc 4.0K 9月 5 10:24 .
drwxrwxr-x 2 ouc ouc 4.0K 9月 5 10:24 2
drwxrwxr-x 2 ouc ouc 4.0K 9月 5 10:24 1
drwxrwxr-x 2 ouc ouc 4.0K 9月 5 09:14 missing
drwxr-xr-x 8 ouc ouc 4.0K 9月 5 09:11 ..
ouc@islouc-vm:~/Desktop/tmp$
```

练习 11:



The image shows a file manager window with a dark theme. The address bar shows 'Desktop' and 'tmp'. The file list contains four items: '1', '2', 'missing', and 'marco.sh'. A preview window for 'marco.sh' is open, showing the following script content:

```
1 #!/bin/bash
2 marco() {
3     echo "$(pwd)" > $HOME/marco_history.log
4     echo "save pwd $(pwd)"
5 }
6 polo() {
7     cd "$(cat "$HOME/marco_history.log")"
8 }
```

Below the file manager is a terminal window with a dark background. It shows the following commands and output:

```
ouc@islouc-vm:~/Desktop/tmp$ ls
1 2 marco.sh missing
ouc@islouc-vm:~/Desktop/tmp$ source marco.sh
ouc@islouc-vm:~/Desktop/tmp$ marco
save pwd /home/ouc/Desktop/tmp
ouc@islouc-vm:~/Desktop/tmp$ cd
ouc@islouc-vm:~$ polo
ouc@islouc-vm:~/Desktop/tmp$
```

## 练习 12:

test.sh

~/Desktop

```

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"

```

rdebug.sh

~/Desktop/tmp

```

1 #!/usr/bin/env bash
2 echo > out.log
3 for ((count=0;;count++))
4 do
5     ./test.sh &>> out.log
6     if [[ $? -ne 0 ]]; then
7         echo "failed after $count times"
8         break
9     fi
10 done&&
11

```

```

ouc@islouc-vm:~/Desktop/tmp$ ls
1 2 debug.sh marco.sh missing test.sh
ouc@islouc-vm:~/Desktop/tmp$ chmod +x debug.sh
ouc@islouc-vm:~/Desktop/tmp$ chmod +x test.sh
ouc@islouc-vm:~/Desktop/tmp$ ./debug.sh
failed after 93 times

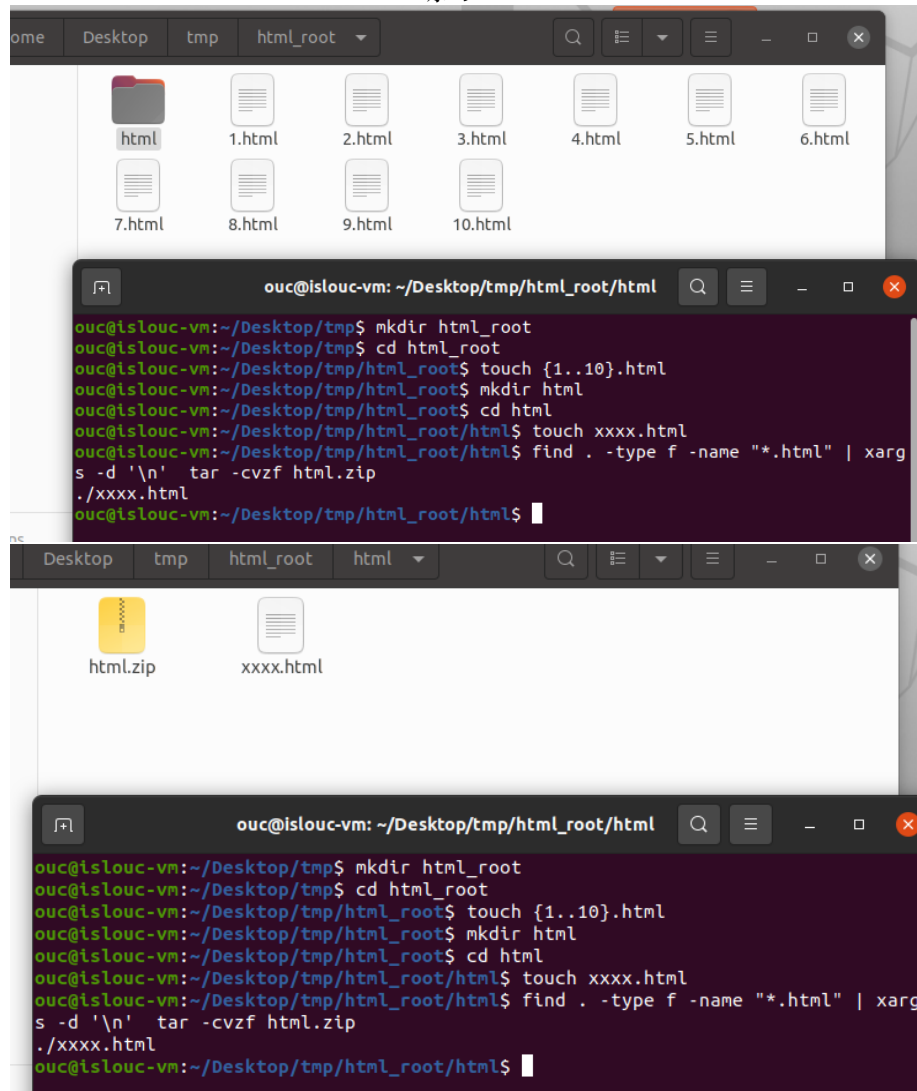
```

```

69 Everything went according to plan
70 Everything went according to plan
71 Everything went according to plan
72 Everything went according to plan
73 Everything went according to plan
74 Everything went according to plan
75 Everything went according to plan
76 Everything went according to plan
77 Everything went according to plan
78 Everything went according to plan
79 Everything went according to plan
80 Everything went according to plan
81 Everything went according to plan
82 Everything went according to plan
83 Everything went according to plan
84 Everything went according to plan
85 Everything went according to plan
86 Everything went according to plan
87 Everything went according to plan
88 Everything went according to plan
89 Everything went according to plan
90 Everything went according to plan
91 Everything went according to plan
92 Everything went according to plan
93 Everything went according to plan
94 Everything went according to plan
95 Something went wrong
96 The error was using magic numbers

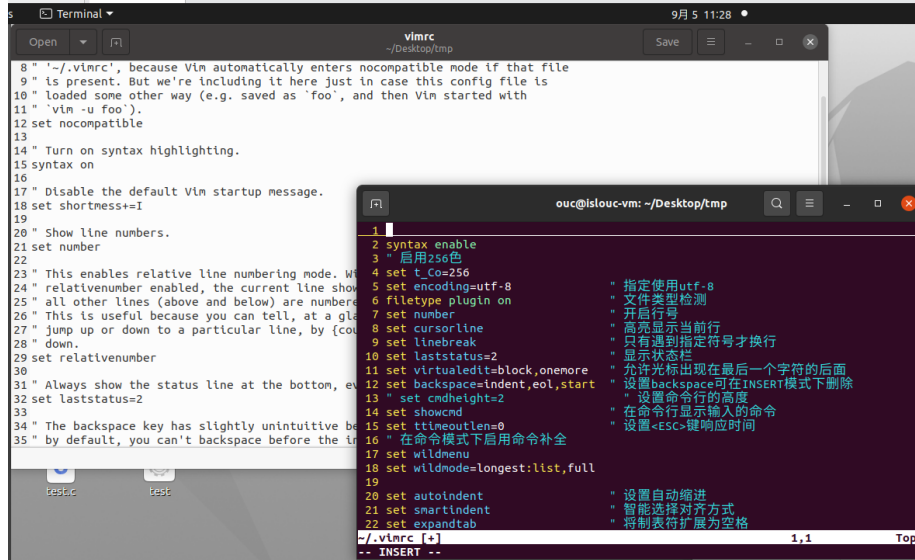
```

### 练习 13:





## 练习 14:



The screenshot shows a terminal window with a dark background. The title bar indicates the window is titled 'vimrc' and the path is '~/Desktop/tmp'. The terminal displays the contents of the vimrc file, which includes comments and configuration settings. The settings are as follows:

```
1 syntax enable
2 " 启用256色
3 set encoding=utf-8
4 filetype plugin on
5 set number
6 set cursorline
7 set linebreak
8 set laststatus=2
9 set virtualedit=block,onemore
10 set backspace=indent,eol,start
11 " set cmdheight=2
12 set showcmd
13 set timeoutlen=0
14 " 在命令模式下启用命令补全
15 set wildmenu
16 set wildmode=longest:list,full
17
18 set autoindent
19 set smartindent
20 set expandtab
```

At the bottom of the terminal, the status line shows '-- INSERT --' and the cursor is at line 1, column 1.

## 练习 15:

5. 将光标下移到第一讲第三节。

### 第一讲第三节：文本编辑之删除

**\*\* 在正常(Normal)模式下，可以按下 x 键来删除光标所在位置的字符。\*\***

1. 请将光标移动到本节中下面标记有 ---> 的那一行。
2. 为了修正输入错误，请将光标移至准备删除的字符的位置处。
3. 然后按下 x 键将错误字符删除掉。
4. 重复步骤2到步骤4，直到句子修正为止。

--> The cow jumped over the moon.

5. 好了，该行已经修正了，下面是第一讲第四节。

特别提示：在浏览本教程时，不要强行记忆。记住一点：在使用中学习。

### 第一讲第四节：文本编辑之插入

**\*\* 在正常模式下，可以按下 i 键来插入文本。\*\***

1. 请将光标移动到本节中下面标记有 ---> 的第一行。
2. 为了使得第一行内容雷同于第二行，请将光标移至文本第一个准备插入字符的位置。
3. 然后按下 i 键，接着输入必要的文本字符。
4. 每个错误修正完毕后，请按下 <ESC> 键返回正常模式。  
重复步骤2至步骤4以便修正句子。

--> There is some text missing from this line .

--> There is some text missing from this line.

5. 如果您对文本插入操作已经很满意，请接着阅读下面的第一讲第五节。

### 第一讲第五节：文本编辑之添加

**\*\* 按 A 键以添加文本。 \*\***

1. 移动光标到下面第一个标记有 ---> 的一行。  
光标放在那一行的哪个字符上并不重要。
2. 按 A 键输入必要的添加内容。
3. 文本添加完后，按 <ESC> 键回到正常模式。
4. 移动光标到下面第二个标记有 ---> 的一行。重复步骤2和步骤3以改正这个句子。

--> There is some text missing from this line.

There is some text missing from this line.

There is also some text missing here.

5. 当您添加文本操作感到满意时，请继续学习第一讲第六节。

- INSERT --

## 练习 16:

3. 请将光标移至准备要删除的单词的起始处。

4. 接着输入 `dw` 删除掉该单词。

特别提示：当您输入时，字母 `d` 会同时出现在屏幕的最后一行。`vim` 在等待您输入字母 `w`。如果您看到的是除 `d` 外的其他字符，那表明您按错了；请按下 `<ESC>` 键，然后重新再来。

```
--> There are some words that don't belong this sentence.
```

5. 重复步骤3和步骤4，直至句子修正完毕。接着继续第二讲第二节内容。

### 第二讲第二节：更多删除类命令

**\*\* 输入 `d$` 从当前光标删除到行末。 \*\***

1. 请按下 `<ESC>` 键确保您处于正常模式。

2. 请将光标移动到本节中下面标记有 `-->` 的那一行。

3. 请将光标移动到该行的尾部(也就是在第一个点号`'.`后面)。

4. 然后输入 `d$` 从光标处删至当前行尾部。

```
--> Somebody typed the end of this line twice.
```

5. 请继续学习第二讲第三节就知道是怎么回事了。

**\*\* 使用操作符时输入数字可以使它重复那么多次。 \*\***

上面已经提到过删除操作符和动作的组合，您可以在组合中动作之前插入一个数字以删除更多：

`d`   `number(数字)`   `motion`

1. 移动光标到下面标记有 `-->` 的一行中第一个大写字母单词上。

2. 输入 `d2w` 以删除两个大写字母单词。

3. 重复步骤1和步骤2，使用不同的数字使得用一个命令就能删除全部相邻的大写字母单词

```
--> this line of words is cleaned up.
```

### 第二讲第六节：操作整行

**\*\* 输入 `dd` 可以删除整个当前行。 \*\***

鉴于整行删除的高频度，`vi` 的设计者决定要简化整行删除操作，您仅需要在同一行上击打两次 `d` 就可以删除掉光标所在的整行了。

1. 请将光标移动到本节中下面的短句段落中的第二行。

2. 输入 `dd` 删除该行。

3. 然后移动到第四行。

4. 接着输入 `2dd` 删除两行。

```
--> 1) Roses are red,  
--> 3) Violets are blue,  
--> 6) Sugar is sweet  
--> 7) And so are you.
```

### 第二讲第七节：撤消类命令

**\*\* 输入 `u` 来撤消最后执行的命令，输入 `U` 来撤消对整行的修改。 \*\***

1. 请将光标移动到本节中下面标记有 `-->` 的那一行，并将其置于第一个错误处。

2. 输入 `x` 删除第一个不想保留的字母。

3. 然后输入 `u` 撤消最后执行的(一次)命令。

4. 这次要使用 `x` 修正本行的所有错误。

5. 现在输入一个大写的 `U`，恢复到该行的原始状态。

6. 接着多次输入 `u` 以撤消 `U` 以及更前的命令。

7. 然后多次输入 `CTRL-R` (先按下 `CTRL` 键不放开，接着按 `R` 键)，这样就可以重做被撤消的命令，也就是撤消掉撤消命令。

```
--> Fix the errors on this line and replace them with undo.
```

8. 这些都是非常有用的命令。下面是第二讲的小结了。

## 练习 17:

### 第三讲第一节：置入类命令

**\*\* 输入 p 将最后一次删除的内容置入光标之后。 \*\***

1. 请将光标移动到本节中下面第一个标记有 ---> 的一行。
2. 输入 dd 将该行删除，这样会将该行保存到 vim 的一个寄存器中。
3. 接着将光标移动到 c) 一行，即准备置入的位置的上方。记住：是上方哦。
4. 然后在正常模式下(<ESC>键进入)输入 p 将该行粘贴置入。
5. 重复步骤2至步骤4，将所有的行依序放置到正确的位置上。

```
--> a) Roses are red,  
--> b) Violets are blue,  
--> c) Intelligence is learned,  
--> d) Can you learn too?
```

### 第三讲第二节：替换类命令

**\*\* 输入 r 和一个字符替换光标所在位置的字符。\*\***

1. 请将光标移动到本节中下面标记有 ----> 的第一行。
2. 请移动光标到第一个出错的位置。
3. 接着输入 r 和要替换成的字符，这样就能将错误替换掉了。
4. 重复步骤2和步骤3，直到第一行已经修改完毕。

```
--> Whan this line was typed in, someone pressed some wrong keys!  
--> When this line was typed in, someone pressed some wrong keys!
```

5. 然后我们继续学习第三讲第三节。

特别提示：切记您要在使用中学习，而不是在记忆中学习。

### 第三讲第三节：更改类命令

**\*\* 要改变文本直到一个单词的末尾，请输入 ce \*\***

1. 请将光标移动到本节中下面标记有 ---> 的第一行。
2. 接着把光标放在单词 lubw 的字母 u 的位置那里。
3. 然后输入 cw 以及正确的单词(在本例中是输入 ine )。
4. 最后按 <ESC> 键，然后光标定位到下一个错误第一个准备更改的字母处。
5. 重复步骤3和步骤4，直到第一个句子完全雷同第二个句子。

```
--> This line has a few words that need changing using the change operator.  
--> This line has a few words that need changing using the change operator.
```

提示：请注意 ce 命令不仅仅是删除了一个单词，它也让您进入插入模式了。

```
INSERT -->
```

### 第三讲第四节：使用c更改更多

**\*\* 更改类操作符可以与删除中使用的同样的动作配合使用。 \*\***

1. 更改类操作符的工作方式跟删除类是一致的。操作格式是：

```
c    [number]  motion
```

2. 动作参数(motion)也是一样的，比如 w 代表单词，\$代表行末等等。
3. 请将光标移动到本节中下面标记有 ---> 的第一行。
4. 接着将光标移动到第一个错误处。
5. 然后输入 c\$ 使得该行剩下的部分更正得同第二行一样。最后按 <ESC> 键。

```
--> The end of this line needs to be corrected using the c$ command.  
--> The end of this line needs to be corrected using the c$ command.
```

## 练习 18:

### 第四讲第一节：定位及文件状态

**\*\* 输入 CTRL-G 显示当前编辑文件中当前光标所在行位置以及文件状态信息。  
输入大写 G 则直接跳转到文件中的某一指定行。\*\***

提示：切记要先通读本节内容，之后才可以执行以下步骤!!!

1. 按下 CTRL 键不放然后按 g 键。我们称这个键组合为 CTRL-G。您会看到页面最底部出现一个状态信息行，显示的内容是当前编辑的文件名和文件中光标位置。请记住行号，它会在步骤3中用到。

提示：您也许会在屏幕的右下角看到光标位置，这会在 'ruler' 选项设置时发生 (参见 :help 'ruler')

2. 输入大写 G 可以使得当前光标直接跳转到文件最后一行。  
输入 gg 可以使得当前光标直接跳转到文件第一行。
3. 输入您曾停留的行号，然后输入大写 G。这样就可以返回到您第一次按下 CTRL-G 时所在的行了。
4. 如果您觉得没问题的话，请执行步骤1至步骤3的操作进行练习。

### 第四讲第二节：搜索类命令

**\*\* 输入 / 加上一个字符串可以用以在当前文件中查找该字符串。\*\***

1. 在正常模式下输入 / 字符。您此时会注意到该字符和光标都会出现在屏幕底部，这跟 : 命令是一样的。
2. 接着输入 errroor <回车>。那个 errroor 就是您要查找的字符串。
3. 要查找同上一轮的字符串，只需要按 n 键。要向相反方向查找同上一轮的字符串，请输入大写 N 即可。
4. 如果您想逆向查找字符串，请使用 ? 代替 / 进行。
5. 要回到您之前的位置按 CTRL-O (按住 Ctrl 键不放同时按下字母 o)。重复按可以回退更多步。CTRL-I 会跳转到较新的位置。

---> "errroor" is not the way to spell error; errroor is an error.  
提示：如果查找已经到达文件末尾，查找会自动从文件头部继续查找，除非 'wrapscan' 选项被复位。

### 第四讲第三节：配对括号的查找

/errroor

### 第四讲第三节：配对括号的查找

**\*\* 输入 % 可以查找配对的括号 )、]、}。 \*\***

1. 把光标放在本节下面标记有 ---> 那一行中的任何一个 (、[ 或 { 处。
2. 接着按 % 字符。
3. 此时光标的位置应当是在配对的括号处。
4. 再次按 % 就可以跳回配对的第一个括号处。
5. 移动光标到另一个 (、)、[、]、{ 或 } 处，按 % 查看其所作所为。

---> This ( is a test line with ('s, ['s ] and {'s } in it. ))

提示：在程序调试时，这个功能用来查找不配对的括号是很有用的。

### 第四讲第四节：替换命令

**\*\* 输入 :s/old/new/g 可以替换 old 为 new。 \*\***

1. 请将光标移动到本节中下面标记有 ---> 的那一行。
2. 输入 :s/thee/the <回车>。请注意该命令只改变光标所在行的第一个匹配串。

## 练习 19:

Exercise 2: Matching With Wildcards

Task	Text	
Match	cat.	✓
Match	896	✓
Match	?=+	✓
Skip	abc1	



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 3: Matching Characters

Task	Text	
Match	can	✓
Match	man	✓
Match	fan	✓
Skip	dan	
Skip	ran	
Skip	pan	



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 4: Excluding Characters

Task	Text	
Match	hog	✓
Match	dog	✓
Skip	bog	



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 5: Matching Character Ranges

Task	Text	
Match	A-na	✓
Match	E-ob	✓
Match	C-pc	✓
Skip	a-ax	
Skip	b-by	
Skip	c-cz	



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 6: Matching Repeated Characters

Task	Text	
Match	wazzzzup	✓
Match	wazzup	✓
Skip	wazup	



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 7: Matching Repeated Characters

Task	Text	
Match	aaaabccc	✓
Match	aabbbbbc	✓
Match	aaacc	✓
Skip	a	



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 1: Matching Characters

Task	Text	
Match	abcdefg	✓
Match	abcde	✓
Match	abc	✓



Solve the above task to continue on to the next problem, or read the [Solution](#).

Exercise 1½: Matching Digits

Task	Text	
Match	abc123xyz	✓
Match	define "123"	✓
Match	var g = 123;	✓



Solve the above task to continue on to the next problem, or read the [Solution](#).

## 练习 20:

Exercise 8: Matching Optional Characters

Task	Text	
Match	1 file found?	✓
Match	2 files found?	✓
Match	24 files found?	✓
Skip	No files found.	

Continue >

Exercise 14: Matching Conditional Text

Task	Text	
Match	I love cats	✓
Match	I love dogs	✓
Skip	I love logs	
Skip	I love cogs	

Continue >

Exercise 15: Matching Other Special Characters

Task	Text	
Match	The quick brown fox jumps over the lazy dog.	✓
Match	There were 614 instances of students getting 90.0% or above.	✓
Match	The FCC had to censor the network for saying &\$#@!.	✓

Continue >

Exercise 11: Matching Groups

Task	Text	Capture Groups
Capture	file_record_transcript.pdf	file_record_transcript ✓
Capture	file_07241999.pdf	file_07241999 ✓
Skip	testfile_fake.pdf.tmp	

Continue >

Exercise 12: Matching Nested Groups

Task	Text	Capture Groups
Capture	Jan 1987	Jan 1987 ✓
Capture	May 1969	May 1969 ✓
Capture	Aug 2011	Aug 2011 ✓

Continue >

Exercise 9: Matching Whitespaces

Task	Text	
Match	1. abc	✓
Match	2. abc	✓
Match	3. abc	✓
Skip	4.abc	

Continue >

Exercise 10: Matching Lines

Task	Text	
Match	Mission: successful	✓
Skip	Last Mission: unsuccessful	
Skip	Next Mission: successful upon capture of target	

Continue >

Exercise 13: Matching Nested Groups

Task	Text	Capture Groups
Capture	1280x720	1280 720 ✓
Capture	1920x1600	1920 1600 ✓
Capture	1024x768	1024 768 ✓

Continue >

## 3 体会与收获

通过本次实验我基本掌握了 SHELL 的用法，明白了在没有图形化之前，我们对电脑的操作方式，尤其是各种各样类型的文字编辑器，特别是学习了 Vim 之后，Vim 自带的教学很清晰明了，几乎涵盖了 Vim 的所有操作。我还有收获就是权限的管理是非常重要的，在运行各种不同的脚本前一定要检查权限，前面做相关练习的时候想到了，但一到了后面运行脚本的时候就忘了，看到报错提示才反应过来。创建 marco 和 polo 函数让我认识到了封装代码的好处，对于脚本错误的处理和监视也同样很有意思。我还学习了正则表达式的运用。

遇到最大的问题是我想要把练习涉及的代码从 linux 系统上移到 windows 上，但是系统不互通，我把一整个文件夹直接 copy 过来会卡死，配置了很久 Samaba 但没有用，后面是一个个文件 copy 过来就可以了。

github 地址:<https://github.com/HollowWarlock/ouc-xitonggongjukaifajichu.git>