

# 系统基础开发工具

## 实验二

姓 名: 陈 佳 玲  
学 号: 23100021002  
专 业: 23 级环境工程  
2025 年 9 月 11 日

## 目录

<b>1</b>	<b>实验目的</b>	<b>3</b>
<b>2</b>	<b>实验内容</b>	<b>3</b>
2.1	Shell 工具和脚本 . . . . .	3
2.1.1	目录显示以及创建文件夹 . . . . .	3
2.1.2	打印输出内容文本 . . . . .	4
2.1.3	环境变量 PATH . . . . .	4
2.1.4	打印当前目录并更改 . . . . .	5
2.1.5	创建新文件并删除 . . . . .	5
2.1.6	查看用户手册 . . . . .	7
2.1.7	查看和连接文件 . . . . .	9
2.1.8	创建脚本并添加内容 . . . . .	9
2.1.9	运行脚本文件并使用 chmod 指令添加权限 . . . . .	10
2.1.10	编写 bash 函数并使用 . . . . .	10
2.1.11	压缩 . . . . .	11
2.1.12	df 查看磁盘空间使用情况 . . . . .	12
2.1.13	进程 . . . . .	13
2.2	编辑器 Vim . . . . .	15
2.2.1	三种模式 . . . . .	15
2.2.2	创建 my.txt 文件 . . . . .	16
2.2.3	编辑模式 . . . . .	16
2.2.4	文档保存 . . . . .	17

2.2.5	打开指定文件并高亮显示关键词 . . . . .	18
2.2.6	调用外部命令 . . . . .	19
2.2.7	替换 . . . . .	21
<b>3</b>	<b>心得体会</b>	<b>23</b>
<b>4</b>	<b>实验代码查看链接</b>	<b>23</b>

## 1 实验目的

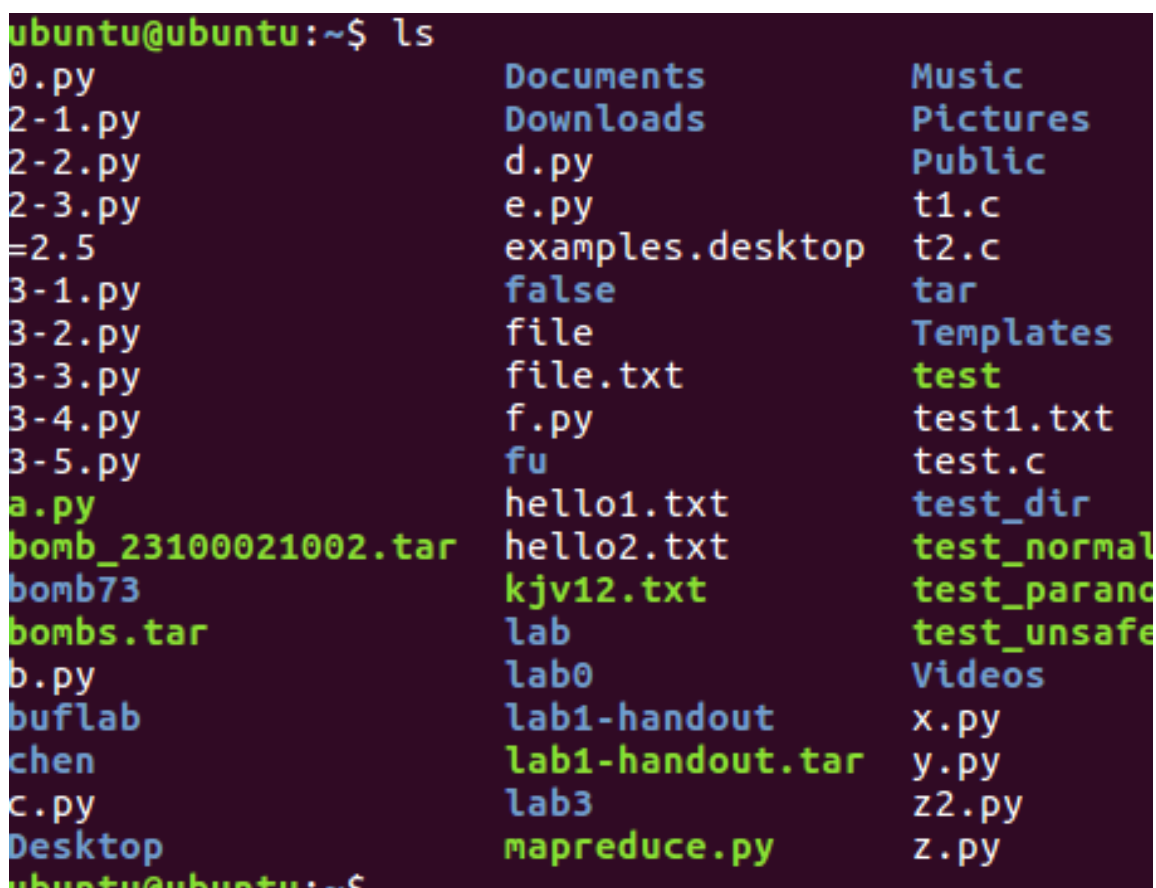
1. 学习 Shell 工具和脚本：通过课堂学习和课下探索，熟悉 shell 基本指令，认识脚本以及一些基本语法知识，强化对于 bash 命令的熟练度，掌握在 Linux 环境下使用 Shell 进行基本操作和自动化任务的能力。
2. 掌握编辑器 Vim 的使用：通过实际操作，学习 Vim 的基本操作模式、常用命令和高级功能，能够熟练使用 Vim 进行文本编辑和代码编写，提高在命令行环境下的编辑效率。
3. 培养系统开发基础能力：通过 Shell 脚本编写和 Vim 编辑器使用的结合，培养在 Unix/Linux 系统环境下进行软件开发的基础能力，为后续课程和项目开发打下坚实基础。

## 2 实验内容

### 2.1 Shell 工具和脚本

#### 2.1.1 目录显示以及创建文件夹

(1) 显示目录：ls



```
ubuntu@ubuntu:~$ ls
0.py           Documents      Music
2-1.py         Downloads     Pictures
2-2.py         d.py          Public
2-3.py         e.py          t1.c
=2.5          examples.desktop t2.c
3-1.py         false         tar
3-2.py         file          Templates
3-3.py         file.txt      test
3-4.py         f.py          test1.txt
3-5.py         fu            test.c
a.py           hello1.txt    test_dir
bomb_23100021002.tar hello2.txt    test_normal
bomb73         kjv12.txt     test_paranc
bombs.tar     lab           test_unsafe
b.py          lab0          Videos
buflab        lab1-handout  x.py
chen          lab1-handout.tar y.py
c.py          lab3          z2.py
Desktop       mapreduce.py  z.py
ubuntu@ubuntu:~$
```

图 1: 目录显示

(2) 创建文件夹 mkdir

```
ubuntu@ubuntu:~$ mkdir aaa.c
ubuntu@ubuntu:~$ ls
Floppy Disk  Documents  Pictures
2-1.py      Downloads  Public
2-2.py      d.py       t1.c
2-3.py      e.py       t2.c
=2.5        examples.desktop tar
3-1.py      false      Templates
3-2.py      file       test
3-3.py      file.txt   test1.txt
3-4.py      f.py       test.c
3-5.py      fu         test_dir
aaa.c       hello1.txt test_normal
a.py        hello2.txt test_paranoic
bomb_23100021002.tar kjv12.txt  test_unsafe
bomb73      lab        Videos
bombs.tar  lab0       x.py
b.py        lab1-handout y.py
buflab      lab1-handout.tar z2.py
chen        lab3       z.py
c.py        mapreduce.py
Desktop     Music
```

图 2: 创建文件夹并验证

### 2.1.2 打印输出内容文本

使用 echo 指令输出文本内容或写入

```
ubuntu@ubuntu:~$ echo hello
hello
ubuntu@ubuntu:~$
```

图 3: echo 打印 hello

### 2.1.3 环境变量 PATH

使用 echo 指令打印 PATH 环境变量, 使用 which echo 可以看到 echo 所在路径

```
ubuntu@ubuntu:~$ echo $PATH
/home/ubuntu/bin:/home/ubuntu/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
ubuntu@ubuntu:~$ which echo
/bin/echo
ubuntu@ubuntu:~$ /bin/echo $PATH
/home/ubuntu/bin:/home/ubuntu/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
ubuntu@ubuntu:~$
```

图 4: 查看 PATH

#### 2.1.4 打印当前目录并更改

1. 打印当前目录: `pwd`
2. 更改当前目录: `cd`

```
ubuntu@ubuntu:~$ pwd
/home/ubuntu
ubuntu@ubuntu:~$ cd /home
ubuntu@ubuntu:/home$ pwd
/home
ubuntu@ubuntu:/home$ cd ..
ubuntu@ubuntu:/$ pwd
/
ubuntu@ubuntu:/$ cd ./home
ubuntu@ubuntu:/home$ pwd
/home
ubuntu@ubuntu:/home$ cd ubuntu
ubuntu@ubuntu:~$ pwd
/home/ubuntu
ubuntu@ubuntu:~$ ../../bin/echo hello
hello
ubuntu@ubuntu:~$
```

图 5: `pwd` 和 `cd` 指令

#### 2.1.5 创建新文件并删除

使用 `touch` 指令创建一个新的文件 `t111.c`, 使用 `ls` 指令确定已经创建之后, 使用 `rm` 指令删除文件

```

ubuntu@ubuntu:~$ touch t111.c
ubuntu@ubuntu:~$ ls
0.py                Documents           Pictures
2-1.py             Downloads          Public
2-2.py             d.py               t111.c
2-3.py             e.py               t1.c
=2.5               examples.desktop  t2.c
3-1.py             false              tar
3-2.py             file               Templates
3-3.py             file.txt           test
3-4.py             f.py               test1.txt
3-5.py             fu                 test.c
aaa.c              hello1.txt         test_dir
a.py               hello2.txt         test_normal
bomb_23100021002.tar kjv12.txt          test_paranoid
bomb73             lab                test_unsafe
bombs.tar          lab0              Videos
b.py              lab1-handout       x.py
buflab            lab1-handout.tar   y.py
chen              lab3               z2.py
c.py              mapreduce.py        z.py
Desktop           Music

ubuntu@ubuntu:~$ rm t111.c
ubuntu@ubuntu:~$ ls
0.py                Documents           Pictures
2-1.py             Downloads          Public
2-2.py             d.py               t1.c
2-3.py             e.py               t2.c
=2.5               examples.desktop  tar
3-1.py             false              Templates
3-2.py             file               test
3-3.py             file.txt           test1.txt
3-4.py             f.py               test.c
3-5.py             fu                 test_dir
aaa.c              hello1.txt         test_normal
a.py               hello2.txt         test_paranoid
bomb_23100021002.tar kjv12.txt          test_unsafe
bomb73             lab                Videos
bombs.tar          lab0              x.py
b.py              lab1-handout       y.py
buflab            lab1-handout.tar   z2.py
chen              lab3               z.py
c.py              mapreduce.py
Desktop           Music

```

图 6: touch 和 rm 指令

### 2.1.6 查看用户手册

使用 `man+` 所需查看指令或操作，可以获得用户指南。如使用 `man echo` 查看 `echo` 指令用户手册

A terminal window with a dark purple background and green text. The prompt is 'ubuntu@ubuntu:~\$'. The command 'man echo' has been entered and executed. The prompt is now 'ubuntu@ubuntu:~\$' again, indicating the command has completed.

```
ubuntu@ubuntu:~$ man echo
ubuntu@ubuntu:~$
```

图 7: man echo

```
ECHO(1)                                User Commands                                ECHO(1)

NAME
    echo - display a line of text

SYNOPSIS
    echo [SHORT-OPTION...]... [STRING]...
    echo LONG-OPTION

DESCRIPTION
    Echo the STRING(s) to standard output.

    -n      do not output the trailing newline

    -e      enable interpretation of backslash escapes

    -E      disable interpretation of backslash escapes (default)

    --help  display this help and exit

    --version
              output version information and exit

    If -e is in effect, the following sequences are recognized:

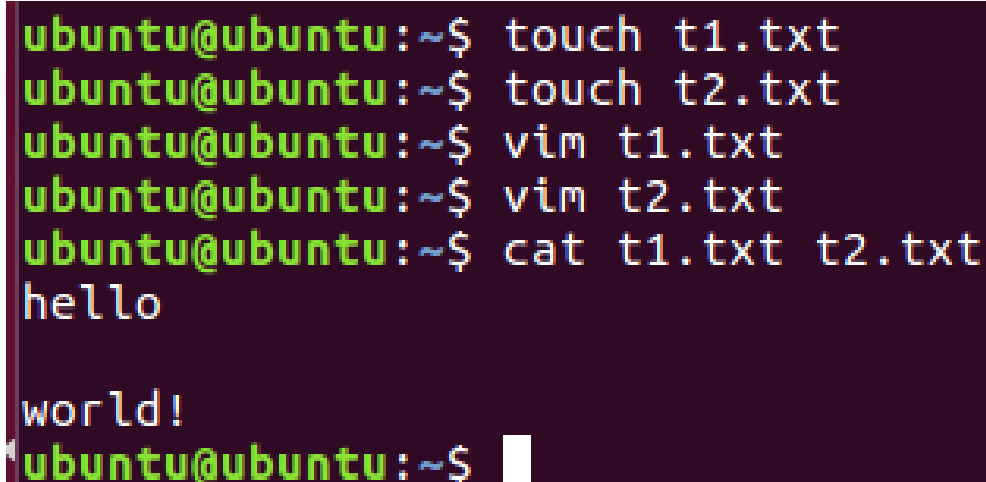
    \\      backslash
    \a      alert (BEL)
    \b      backspace
    \c      produce no further output
    \e      escape
    \f      form feed
    \n      new line
    \r      carriage return
    \t      horizontal tab
```

图 8: man echo 指令结果



### 2.1.7 查看和连接文件

使用 `cat` 查看文件内容、连接多个文件内容到标准输出。如 `cat t1.txt t2.txt` 查看 `t1.txt` 和 `t2.txt` 的内容。



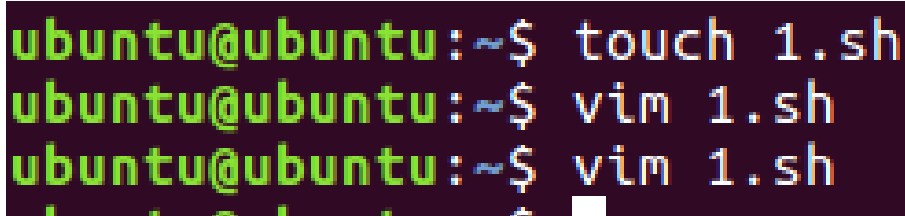
```
ubuntu@ubuntu:~$ touch t1.txt
ubuntu@ubuntu:~$ touch t2.txt
ubuntu@ubuntu:~$ vim t1.txt
ubuntu@ubuntu:~$ vim t2.txt
ubuntu@ubuntu:~$ cat t1.txt t2.txt
hello
world!
ubuntu@ubuntu:~$
```

图 9: `cat` 指令

### 2.1.8 创建脚本并添加内容

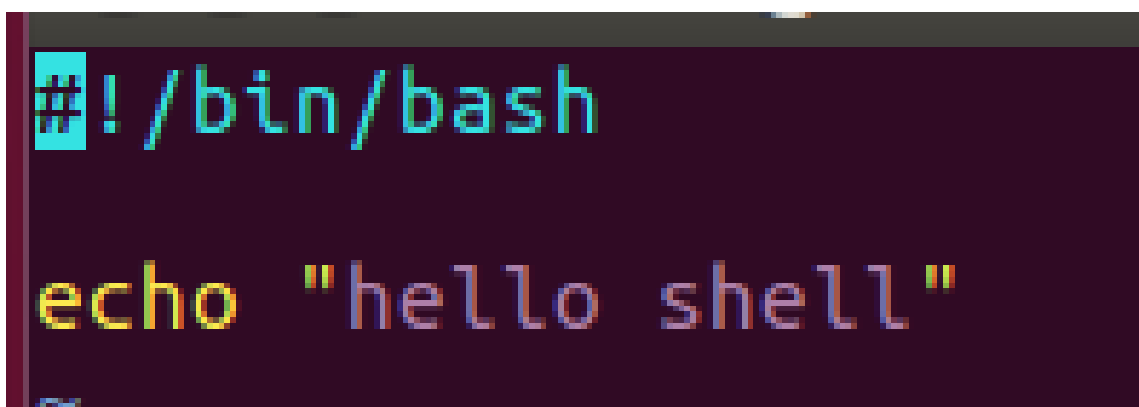
使用指令 `touch 1.sh` 创建一个空的脚本文件，并使用 `vim` 编辑内容。在脚本中要把 `shell` 命令放到一个“脚本”当中，有一个要求：脚本的第一行必须写成类似这样的格式：

```
#!/bin/bash
```



```
ubuntu@ubuntu:~$ touch 1.sh
ubuntu@ubuntu:~$ vim 1.sh
ubuntu@ubuntu:~$ vim 1.sh
ubuntu@ubuntu:~$
```

图 10: 操作



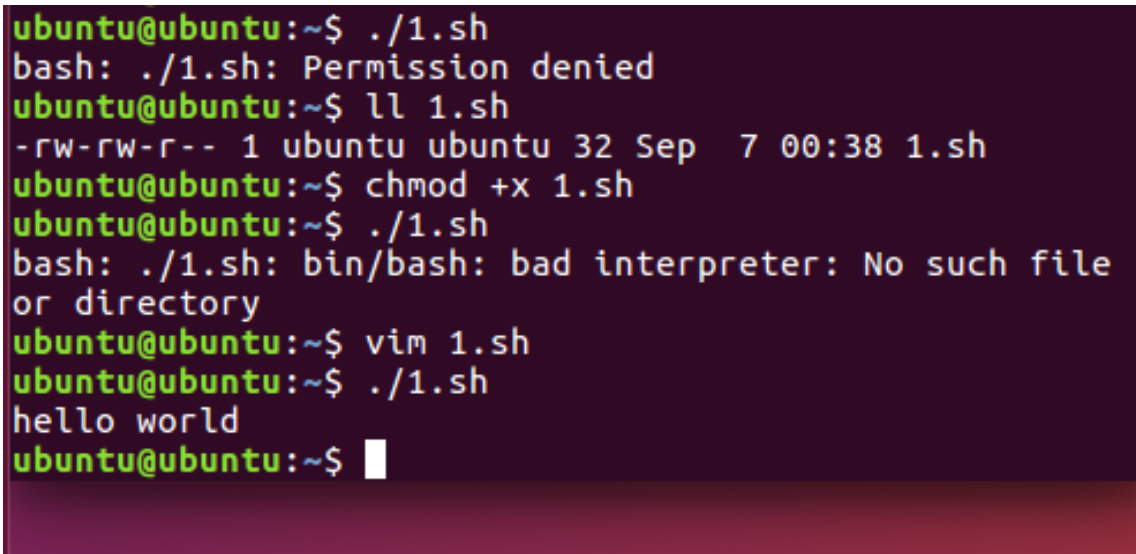
```
#!/bin/bash

echo "hello shell"
```

图 11: 脚本文件内容

### 2.1.9 运行脚本文件并使用 chmod 指令添加权限

运行脚本文件 1.sh 缺少权限，使用 ll 指令查看权限之后，使用 chmod 指令添加权限，运行成功打印出 hello world。



```
ubuntu@ubuntu:~$ ./1.sh
bash: ./1.sh: Permission denied
ubuntu@ubuntu:~$ ll 1.sh
-rw-rw-r-- 1 ubuntu ubuntu 32 Sep  7 00:38 1.sh
ubuntu@ubuntu:~$ chmod +x 1.sh
ubuntu@ubuntu:~$ ./1.sh
bash: ./1.sh: bin/bash: bad interpreter: No such file
or directory
ubuntu@ubuntu:~$ vim 1.sh
ubuntu@ubuntu:~$ ./1.sh
hello world
ubuntu@ubuntu:~$
```

图 12: 脚本运行与 chmod 权限添加

### 2.1.10 编写 bash 函数并使用

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。通过 source 来加载函数，随后可以在 bash 中直接使用。



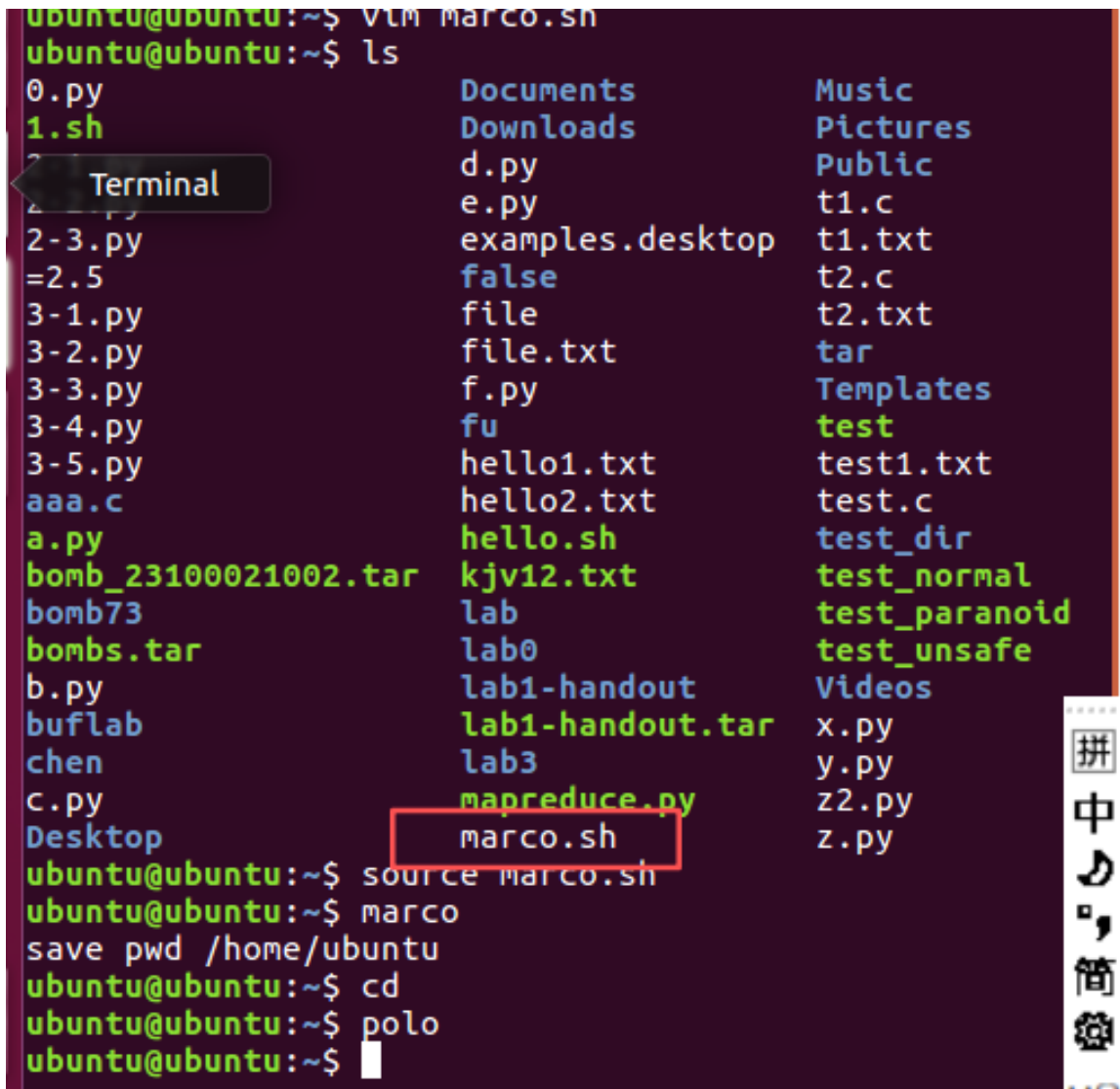
```
#!/bin/bash

marco(){
    echo "$(pwd)" > $HOME/marco_history.log
    echo "save pwd $(pwd)"
}

polo(){
    cd "$(cat "$HOME/marco_history.log")"
}

~
~
~
~
```

图 13: marco.sh



The image shows a terminal window with a dark background. At the top, the prompt is 'ubuntu@ubuntu:~\$'. The user enters 'vfm marco.sh'. The prompt changes to 'ubuntu@ubuntu:~\$'. The user enters 'ls', which lists files in three columns. The first column contains files like '0.py', '1.sh', '2-3.py', etc. The second column contains 'Documents', 'Downloads', 'd.py', etc. The third column contains 'Music', 'Pictures', 'Public', etc. A red box highlights 'marco.sh' in the second column. The user then enters 'source marco.sh', followed by 'marco', 'save pwd /home/ubuntu', 'cd', 'polo', and finally a blank line.

```
ubuntu@ubuntu:~$ vfm marco.sh
ubuntu@ubuntu:~$ ls
0.py          Documents    Music
1.sh          Downloads   Pictures
2-3.py        d.py        Public
=2.5          e.py        t1.c
3-1.py        examples.desktop t1.txt
3-2.py        false       t2.c
3-3.py        file        t2.txt
3-4.py        file.txt    tar
3-5.py        f.py        Templates
aaa.c         fu          test
a.py          hello1.txt  test1.txt
bomb_23100021002.tar kjv12.txt   test.c
bomb73        lab         test_dir
bombs.tar     lab0        test_normal
b.py          lab1-handout test_paranoid
buflab        lab1-handout.tar test_unsafe
chen          lab3        Videos
c.py          mapreduce.py x.py
Desktop       marco.sh     y.py
ubuntu@ubuntu:~$ source marco.sh
ubuntu@ubuntu:~$ marco
save pwd /home/ubuntu
ubuntu@ubuntu:~$ cd
ubuntu@ubuntu:~$ polo
ubuntu@ubuntu:~$
```

图 14: 使用操作

### 2.1.11 压缩

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。



```
ubuntu@ubuntu:~$ mkdir html_root
ubuntu@ubuntu:~$ cd html_root
ubuntu@ubuntu:~/html_root$ touch {1..10}.html
ubuntu@ubuntu:~/html_root$ mkdir html
ubuntu@ubuntu:~/html_root$ cd html
ubuntu@ubuntu:~/html_root/html$ touch chen.html
ubuntu@ubuntu:~/html_root/html$ find . -type f -name "*.html" | xargs -d '\n' tar -cvzf html.zip
./chen.html
ubuntu@ubuntu:~/html_root/html$ cd chen.html
bash: cd: chen.html: Not a directory
ubuntu@ubuntu:~/html_root/html$ pwd html.root
/home/ubuntu/html_root/html
ubuntu@ubuntu:~/html_root/html$
```

图 15: 操作

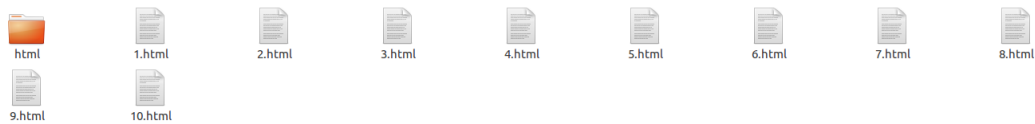


图 16: 创建的 html 文件

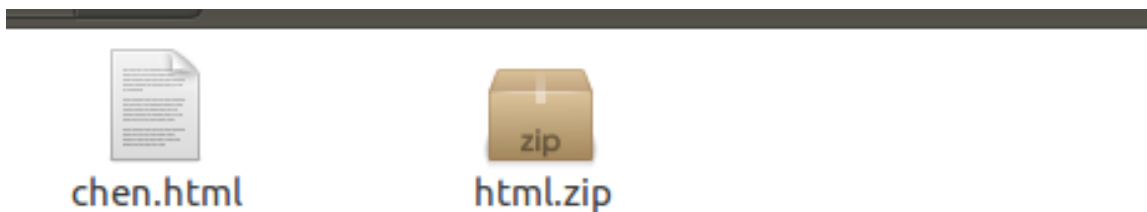
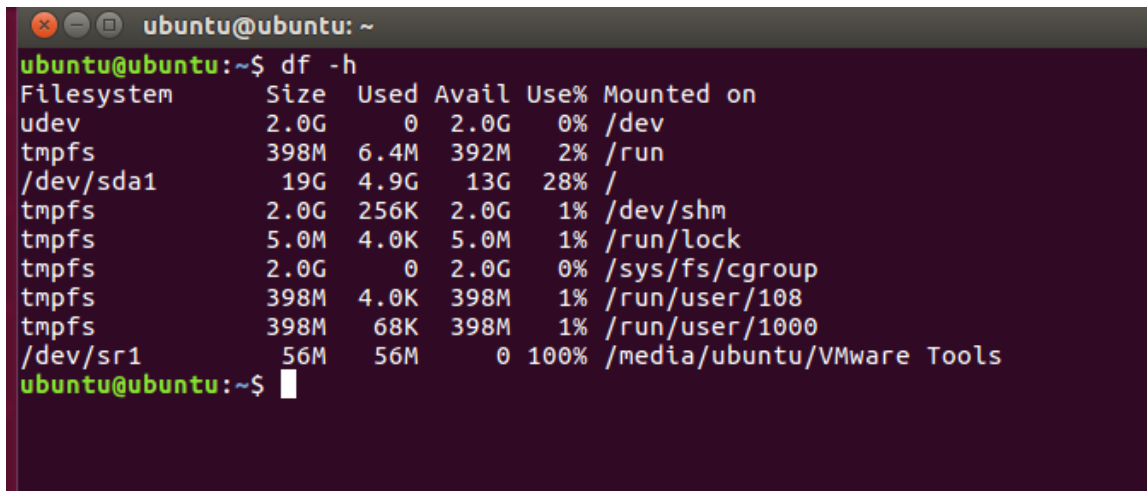


图 17: 压缩结果

#### 2.1.12 df 查看磁盘空间使用情况

df 指令，显示文件系统的磁盘空间使用情况，监控磁盘空间。



```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ df -h  
Filesystem      Size  Used Avail Use% Mounted on  
udev            2.0G   0    2.0G   0% /dev  
tmpfs           398M  6.4M  392M   2% /run  
/dev/sda1       19G   4.9G   13G  28% /  
tmpfs           2.0G  256K   2.0G   1% /dev/shm  
tmpfs           5.0M   4.0K   5.0M   1% /run/lock  
tmpfs           2.0G   0    2.0G   0% /sys/fs/cgroup  
tmpfs           398M   4.0K  398M   1% /run/user/108  
tmpfs           398M   68K  398M   1% /run/user/1000  
/dev/sr1        56M   56M    0 100% /media/ubuntu/VMware Tools  
ubuntu@ubuntu:~$
```

图 18: 查看磁盘空间使用情况

### 2.1.13 进程

(1) ps - 查看活动进程，显示当前系统中的活动进程，能够监控和管理进程。如，ps aux 显示系统中所有进程的详细列表。

```
ubuntu@ubuntu:~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	1.8	0.1	24136	5020	?	Ss	07:58	0:08	/sbin/init a
root	2	0.0	0.0	0	0	?	S	07:58	0:00	[kthreadd]
root	4	0.0	0.0	0	0	?	I<	07:58	0:00	[kworker/0:0
root	6	0.0	0.0	0	0	?	I<	07:58	0:00	[mm_percpu_w
root	7	0.1	0.0	0	0	?	S	07:58	0:00	[ksoftirqd/0
root	8	0.2	0.0	0	0	?	I	07:58	0:00	[rcu_sched]
root	9	0.0	0.0	0	0	?	I	07:58	0:00	[rcu_bh]
root	10	0.0	0.0	0	0	?	S	07:58	0:00	[migration/0
root	11	0.0	0.0	0	0	?	S	07:58	0:00	[watchdog/0]
root	12	0.0	0.0	0	0	?	S	07:58	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	07:58	0:00	[cpuhp/1]
root	14	0.0	0.0	0	0	?	S	07:58	0:00	[watchdog/1]
root	15	0.0	0.0	0	0	?	S	07:58	0:00	[migration/1
root	16	0.2	0.0	0	0	?	S	07:58	0:00	[ksoftirqd/1
root	18	0.0	0.0	0	0	?	I<	07:58	0:00	[kworker/1:0
root	19	0.0	0.0	0	0	?	S	07:58	0:00	[kdevtmpfs]
root	20	0.0	0.0	0	0	?	I<	07:58	0:00	[netns]
root	21	0.0	0.0	0	0	?	S	07:58	0:00	[rcu_tasks_k
root	22	0.0	0.0	0	0	?	S	07:58	0:00	[kauditd]
root	23	0.0	0.0	0	0	?	I	07:58	0:00	[kworker/0:1
root	24	0.0	0.0	0	0	?	S	07:58	0:00	[khungtaskd]
root	25	0.0	0.0	0	0	?	S	07:58	0:00	[oom_reaper]
root	26	0.0	0.0	0	0	?	I<	07:58	0:00	[writeback]
root	27	0.0	0.0	0	0	?	S	07:58	0:00	[kcompactd0]
root	28	0.0	0.0	0	0	?	SN	07:58	0:00	[ksmd]
root	29	0.0	0.0	0	0	?	SN	07:58	0:00	[khugepaged]
root	30	0.0	0.0	0	0	?	I<	07:58	0:00	[crypto]
root	31	0.0	0.0	0	0	?	I<	07:58	0:00	[kintegrityd
root	32	0.0	0.0	0	0	?	I<	07:58	0:00	[kblockd]
root	33	0.1	0.0	0	0	?	I	07:58	0:00	[kworker/1:1
root	34	0.0	0.0	0	0	?	I<	07:58	0:00	[ata_sff]
root	35	0.0	0.0	0	0	?	I<	07:58	0:00	[md]
root	36	0.0	0.0	0	0	?	I<	07:58	0:00	[edac-poller
root	37	0.0	0.0	0	0	?	I<	07:58	0:00	[devfreq_wq]
root	38	0.0	0.0	0	0	?	I<	07:58	0:00	[watchdogd]
root	41	0.0	0.0	0	0	?	S	07:58	0:00	[kswapd0]

图 19: ps - 查看活动进程

(2) top - 实时显示进程及系统资源的使用情况, 动态监控系统进程状态。如, 直接运行 top 会打开一个交互界面, 显示当前活动进程及资源使用情况。

```
top - 08:06:28 up 7 min, 1 user, load average: 0.09, 0.61, 0.46
Tasks: 231 total, 1 running, 160 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.2 us, 1.0 sy, 0.0 ni, 96.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4070380 total, 2894284 free, 439572 used, 736524 buff/cache
KiB Swap: 998396 total, 998396 free, 0 used, 3221000 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1008	root	20	0	207104	54700	32756	S	2.3	1.3	0:12.70	Xorg
2192	ubuntu	20	0	272436	88576	61192	S	2.0	2.2	0:11.95	compiz
2493	ubuntu	20	0	119308	31244	26424	S	1.0	0.8	0:01.59	gnome-ter
2240	ubuntu	20	0	105592	30628	27132	S	0.7	0.8	0:02.12	vmtoolsd
370	root	20	0	52388	8784	7848	S	0.3	0.2	0:02.92	vmtoolsd
3160	ubuntu	20	0	8096	3472	2920	R	0.3	0.1	0:00.22	top
1	root	20	0	24136	5020	3712	S	0.0	0.1	0:08.23	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu
7	root	20	0	0	0	0	S	0.0	0.0	0:00.69	ksoftirqd
8	root	20	0	0	0	0	I	0.0	0.0	0:01.00	rcu_sched
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.03	migration
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.07	migration
16	root	20	0	0	0	0	S	0.0	0.0	0:00.93	ksoftirqd
18	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1
19	root	20	0	0	0	0	S	0.0	0.0	0:00.07	kdevtmpfs
20	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
Terminal	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks
23	root	20	0	0	0	0	I	0.0	0.0	0:00.02	kauditd
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kworker/0
25	root	20	0	0	0	0	S	0.0	0.0	0:00.00	oom_reape
26	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	writeback
27	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kcompactd
28	root	25	5	0	0	0	S	0.0	0.0	0:00.00	ksmd
29	root	39	19	0	0	0	S	0.0	0.0	0:00.00	khugepage
30	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	crypto
31	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kintegrit
32	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kblockd
33	root	20	0	0	0	0	I	0.0	0.0	0:00.47	kworker/1
34	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	ata_sff

图 20: top - 实时显示进程动态

## 2.2 编辑器 Vim

### 2.2.1 三种模式

基本上 vi/vim 共分为三种模式

1. 命令模式：用户刚刚启动 vi/vim，便进入了命令模式。此状态下敲击键盘动作会被 Vim 识别为命令，而非输入字符，比如我们此时按下 i，并不会输入一个字符，i 被当作了一个命令。
2. 输入模式：在命令模式下按下 i 就进入了输入模式，使用 Esc 键可以返回到普通模式。
3. 命令行模式：在命令模式下按下:（英文冒号）就进入了底线命令模式。底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。按 ESC 键可随时退出底线命令模式。

### 2.2.2 创建 my.txt 文件

输入命令 `vi+ 文件名` 便可创建文件并进入 vi 模式编辑器当中

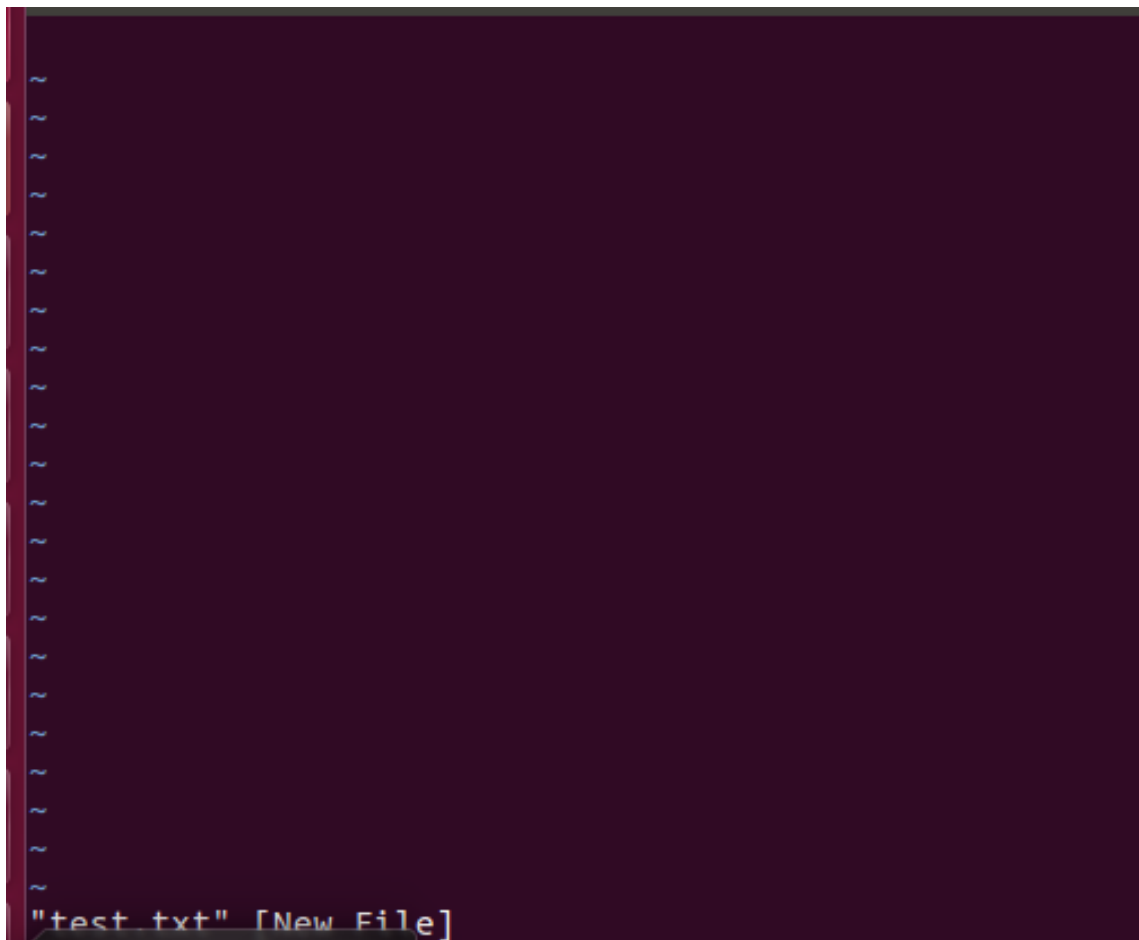


图 21: vi 编辑器页面

### 2.2.3 编辑模式

在一般模式之中，只要按下 `i`, `o`, `a` 等字符就可以进入输入模式了。在编辑模式当中，你可以发现在左下角状态栏中会出现 `-INSERT-` 的字样，那就是可以输入任意字符的提示。这个时候，键盘上除了 `Esc` 这个按键之外，其他的按键都可以视作为一般的输入按钮了，所以你可以进行任何的编辑。



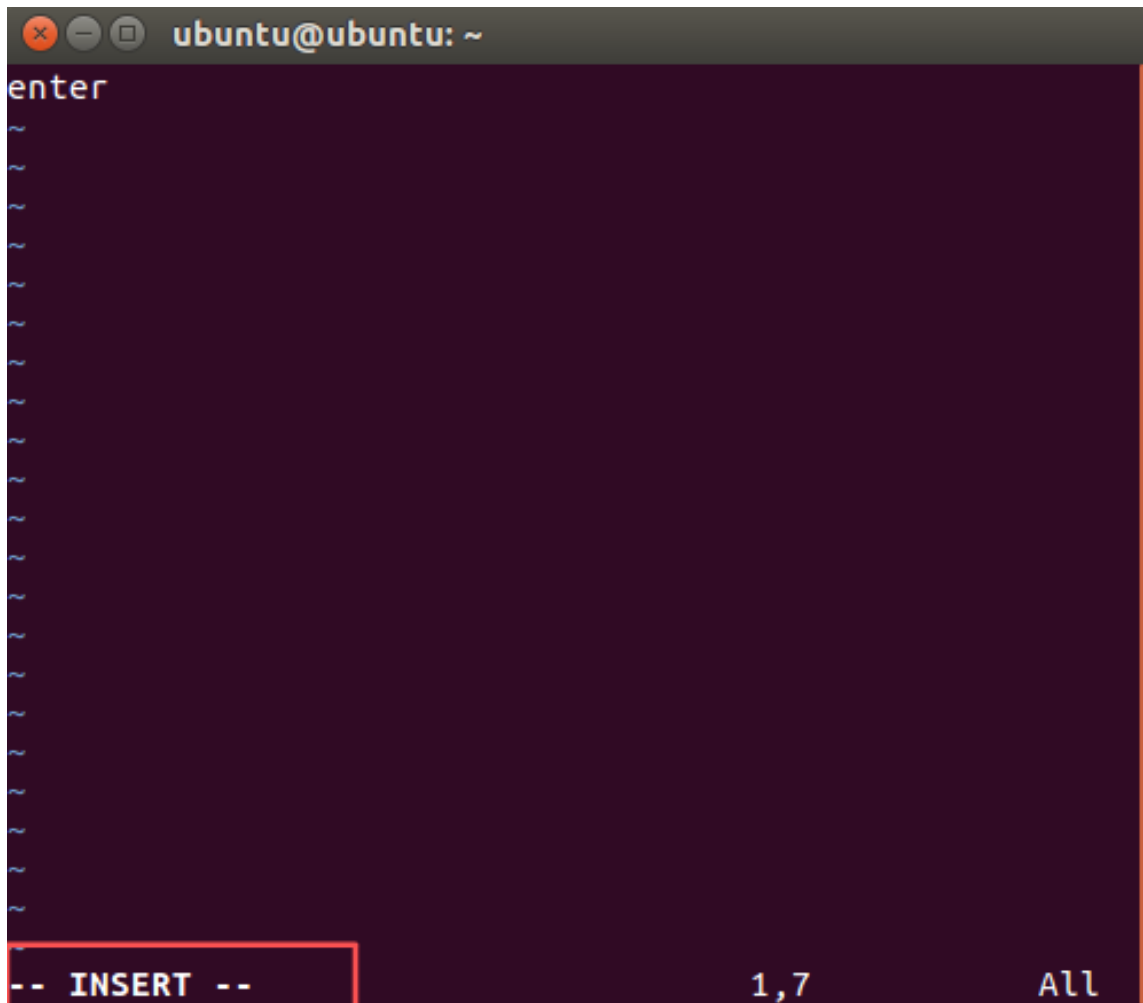


图 22: 编辑模式

#### 2.2.4 文档保存

在一般模式中按下:wq 储存后离开 vi

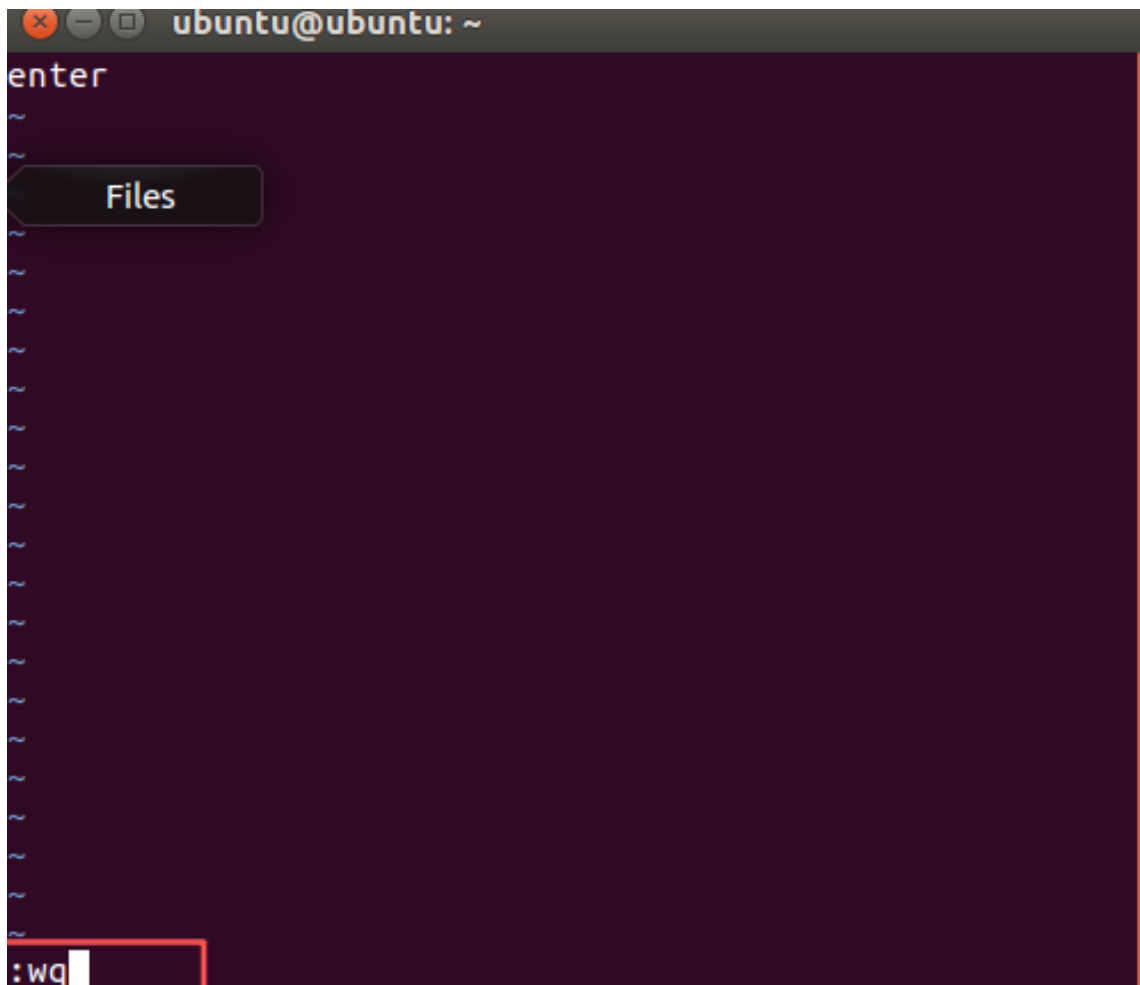


图 23: 退出并保存

### 2.2.5 打开指定文件并高亮显示关键词

使用 `vim +/关键词 + 文件路径`，打开指定文件并高亮显示关键词。如 `vim +/echo 1.sh`

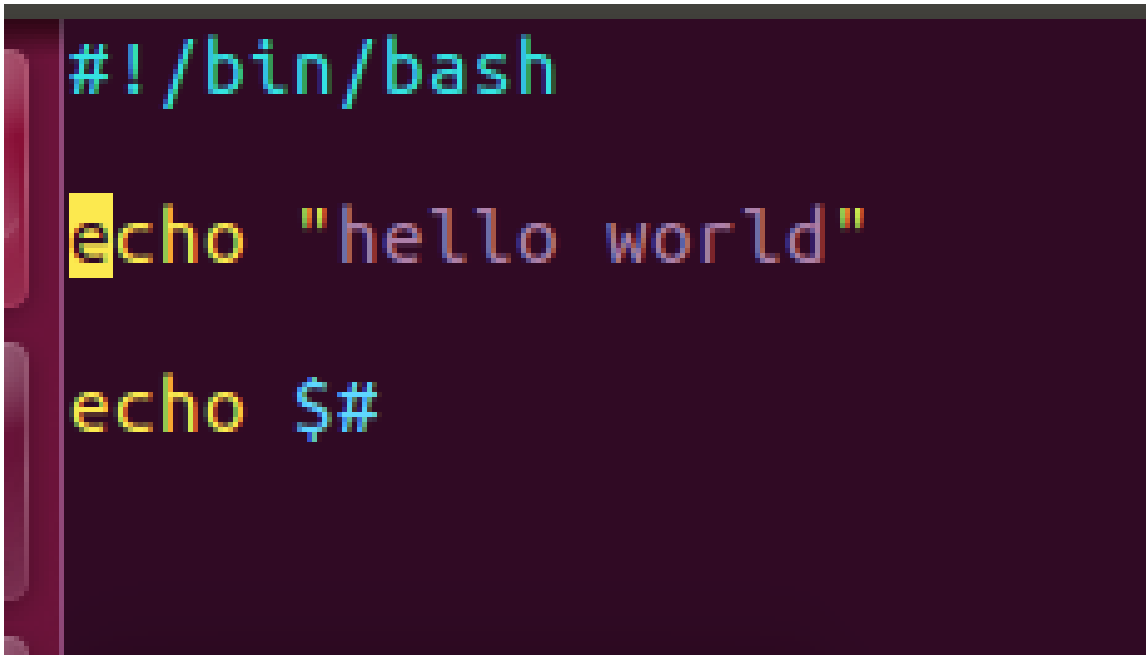


图 24: 查看高亮结果

|

### 2.2.6 调用外部命令

切换到底行模式下，输入:!  
和外部命令，比如说外部命令 ls, 就是:ls

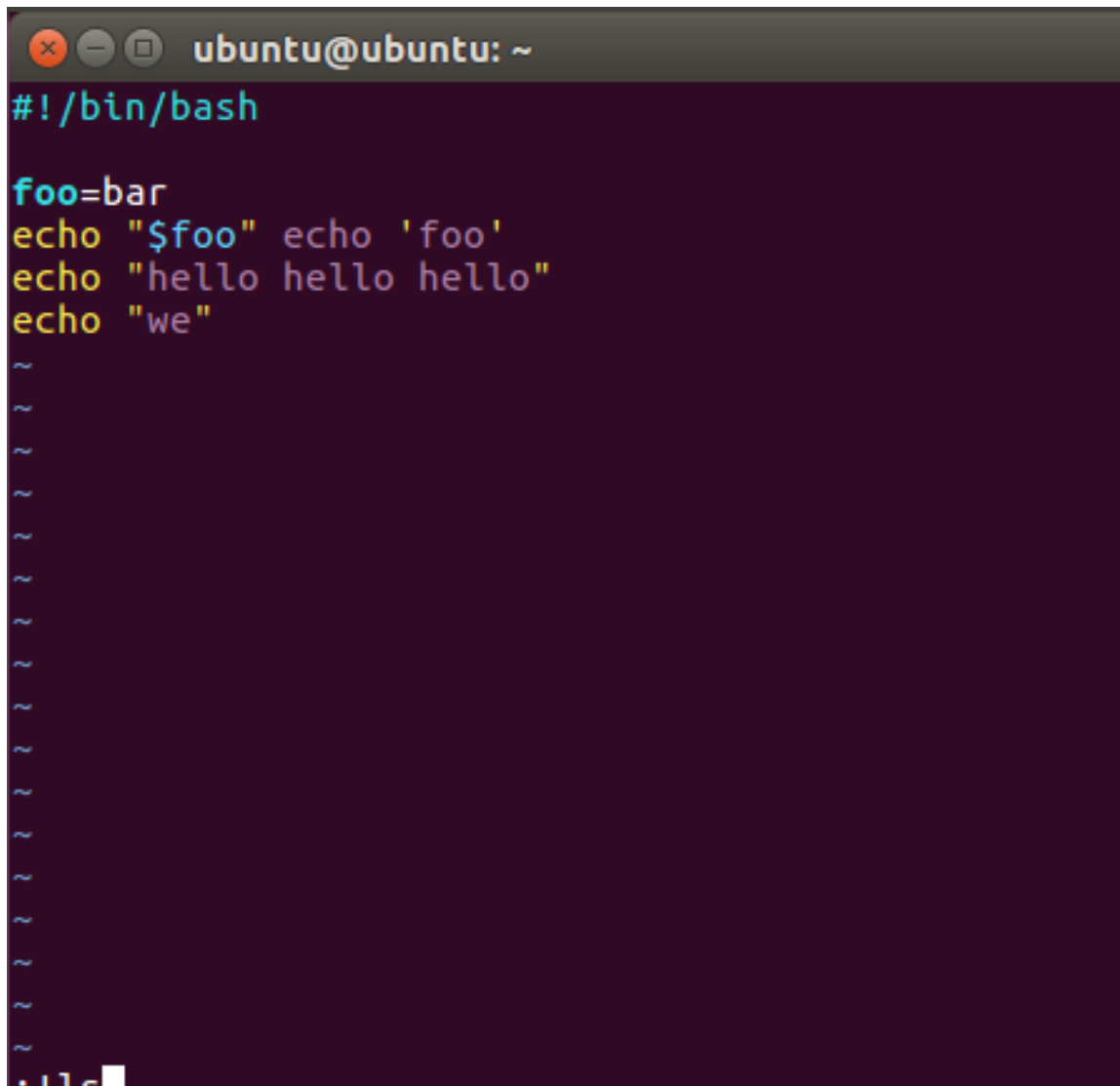


图 25: 调用外部命令

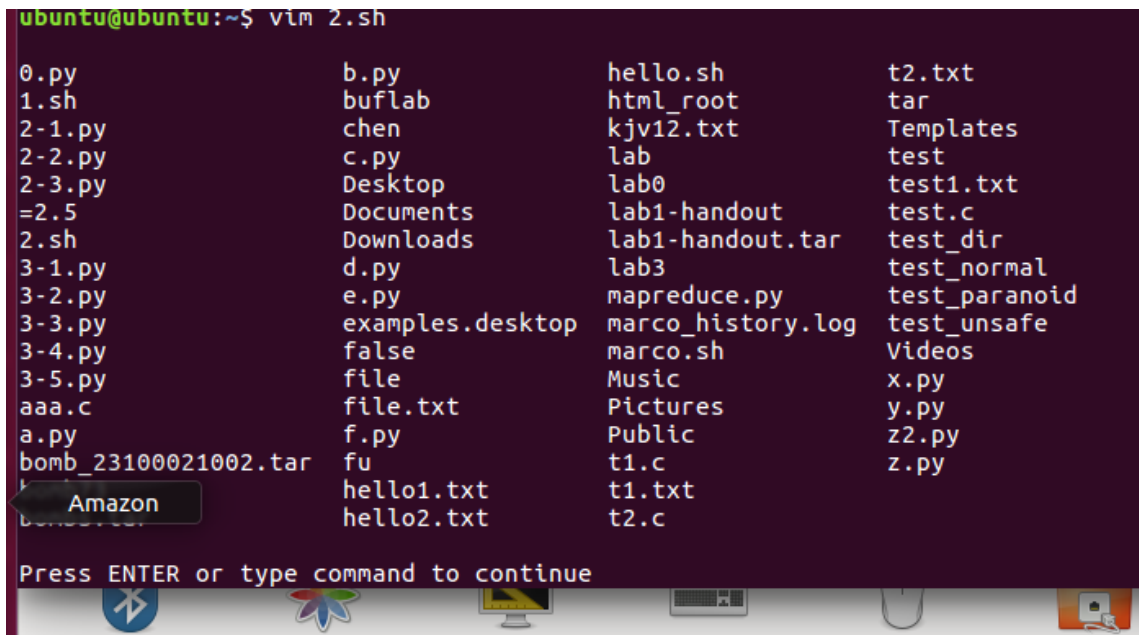


图 26: 外部命令调用结果

### 2.2.7 替换

切换到底行模式下，输入`%s/string1/string2/g`，把 `string1` 替换成 `string2`，下面把 `hello` 替换为 `world`



|

### 3 心得体会

通过本次实验，我实现了从理论到实践的跨越，深刻体会到 Shell 脚本与 Vim 编辑器在 Linux 环境下的核心。实验使我从零散命令的使用，到学会了用 Shell 脚本自动化复杂任务。同时，我从最初不适应 Vim 的模式切换，到熟练运用其命令完成快速编辑、查找和宏操作，深刻理解了 Vim 区分正常模式、插入模式和可视模式的设计哲学，切身感受到了其高效的编辑哲学。此次实验不仅是工具学习，更是一次思维训练，为我后续课程和项目开发提供了至关重要的底层技能与信心。

### 4 实验代码查看链接

本次报告相关练习、报告和代码均可以在 <https://github.com/chen2-spec/my-latex-report> 查看