

系统开发工具基础课程实验报告

姓名：张誉馨

2024 年 9 月 11 日

目录

1	练习内容和结果	1
1.1	Shell工具和脚本.....	1
1.2	编辑器Vim.....	3
1.3	数据整理.....	4
2	解题感悟	4
3	github链接	4

1 练习内容和结果

1.1 Shell工具和脚本

1. 阅读 *man ls* , 然后使用 *ls* 命令进行如下操作:
2. 所有文件 (包括隐藏文件): *-a*
3. 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):

-h

4. 文件以最近访问顺序排序: *-t*
5. 以彩色文本显示输出结果: *--color = auto*

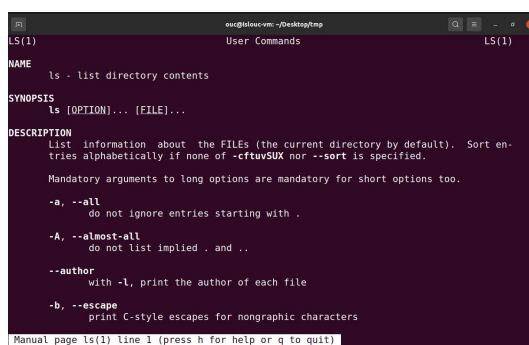


图 1: man ls

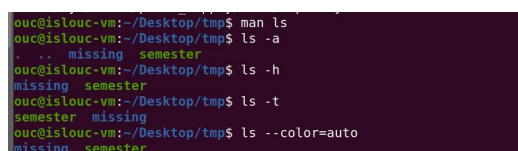


图 2: 使用ls命令进行操作

6. 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。

7. 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

```

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 }

```

图 3: 两个bash函数marco和polo

```

ouc@islouc-vm:~/Desktop/tmp$ vim marco.sh
ouc@islouc-vm:~/Desktop/tmp$ ls
marco.sh  missing  semester
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

```

图 4: source marco.sh 命令

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 echo > out.log
3 for ((count=0;;count++))
4 do
5     ./buggy.sh &>> out.log
6     if [[ $? -ne 0 ]]; then
7         echo "failed after $count times"
8         break
9     fi
10 done
12

```

图 5: 编写的bash脚本

8. 本节课我们讲解的 `find` 命令中的 `-exec` 参数非常强大，它可以对我们查找的文件进行操作。如果我们要对所有文件进行操作呢？例如创建一个zip压缩文件？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，但是有些命令，例如`tar` 则需要从参数接受输入。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（提示：查看 `xargs`的参数-d）译注：MacOS 上的 `xargs`没有-d，查看这个issue

如果您使用的是 MacOS，请注意默认的 BSD `find` 与GNU `coreutils` 中的是不一样的。你可以为`find`添加`-print0`选项，并为`xargs`添加`-0`选项。作为Mac用户，您需要注意 mac 系统自带的命令行工具和 GNU 中对应的工

```
ouc@islouc-vm:~/Desktop/tmp$ vim ./debug_for.sh
ouc@islouc-vm:~/Desktop/tmp$ chmod 777 debug_for.sh
ouc@islouc-vm:~/Desktop/tmp$ ./debug_for.sh
failed after 0 times
```

图 6: 报告脚本在失败前运行多少次

具是有区别的；如果你想使用 GNU 版本的工具，也可以使用 brew 来安装

(1) 首先创建所需的文件

(2) 执行find命令

```
ouc@islouc-vm:~/Desktop/tmp/html_root/html$ mkdir html_root
ouc@islouc-vm:~/Desktop/tmp/html_root/html$ cd html_root
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root$ touch {1..10}.html
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root$ mkdir html
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root$ cd html
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root/html$ touch xxxx.html
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root/html$ find . -type f -name "*.html" |
xargs -d '\n' tar -cvzf html.zip
./xxxx.html
```

图 7: 创建所需的文件后执行find命令

9. (进阶) 编写一个命令或脚本递归的查找文件夹中最近使用的文件。更通用的做法，你可以按照最近的使用时间列出文件吗？

`find . -type f -print0 | xargs -0 ls -lt | head -1`

```
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root/html$ find . -type f -print0 | xargs
0 ls -lt | head
-rw-rw-r-- 1 ouc ouc 115 9月 11 10:24 ./html.zip
-rw-rw-r-- 1 ouc ouc 0 9月 11 10:24 ./xxxx.html
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root/html$ find . -type f -print0 | xargs
0 ls -lt | head -1
-rw-rw-r-- 1 ouc ouc 115 9月 11 10:24 ./html.zip
ouc@islouc-vm:~/Desktop/tmp/html_root/html/html_root/html$ find . -type f -mmin -60 -print0
| xargs 0 ls -lt | head -10
-rw-rw-r-- 1 ouc ouc 115 9月 11 10:24 ./html.zip
-rw-rw-r-- 1 ouc ouc 0 9月 11 10:24 ./xxxx.html
```

图 8: 编写脚本按照最近的使用时间列出文件

1.2 编辑器Vim

1.完成 vimtutor。备注：它在一个 80x24（80 列，24 行）终端窗口看起来效果最好。 vimtutor

```
vim
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 25-30 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 them
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
Hint: The h key is at the left and moves left.
~/tmp/tutor/vim16* 970 lines, 33257 characters
```

图9: vimtutor

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。

```
ooc@islauc-vm:~/Desktop/tmp/html_root/html/html_root/html$ ./var/run/vmblock-fuse/blockdir/
GPDLBA/vimrc
/var/run/vmblock-fuse/blockdir/GPDLBA/vimrc: command substitution: line 3: unexpected EOF w
hile looking for matching '"'
/var/run/vmblock-fuse/blockdir/GPDLBA/vimrc: command substitution: line 4: syntax error: un
expected end of file
/var/run/vmblock-fuse/blockdir/GPDLBA/vimrc: line 3: '$' Comments in Vimscript start with a
'\n\n': command not found
/var/run/vmblock-fuse/blockdir/GPDLBA/vimrc: line 11: syntax error near unexpected token `)'
/var/run/vmblock-fuse/blockdir/GPDLBA/vimrc: line 11: `)' vim -u foo')."
Error detected while processing /home/ooc/.vimrc:
line 34:
E492: Not an editor command: 然后把它保存到 ~/.vimrc! Comments in Vimscript start with a
'"'
Press ENTER or type command to continue
```

图10: 下载vimrc后保存

```
26 " 语法高亮
25 syntax enable
24 " 启用256色
23 set t Co=256
22 set encoding=utf-8          " 指定使用 utf-8
21 filetype plugin on          " 文件类型检测
20 set number                  " 开启行号
19 set cursorline              " 高亮当前行
18 set linebreak               " 只有遇到指定符号才换行
17 set laststatus=2            " 显示状态栏
16 set virtualetit=block,onore " 允许标题出现在最后一个字符的后面
15 set backspace=indent,eol,start " 设置backspace可在INSERT模式下删除
14 set cmdheight=2             " 设置命令行的高度
13 set showcmd                 " 在命令行显示输入的命令
12 set ttimeoutlen=0           " 设置<ESC>键响应时间
11 " 在命令模式下启用命令补全
10 set wildmenu
9 set wildmode=longest,list,full
8
7 set autoindent               " 设置自动缩进
6 set smartindent              " 智能选择对齐方式
5 set expandtab                 " 将制表符扩展为空格
4 set tabstop=4                " 设置tab=4空格
3 set shiftwidth=4             " 设置缩进为4空格
2 set smarttab                 " 在行和段开始处使用制表符
1
27 " 高亮搜索结果
~/.vimrc 27,1 Top
```

图11: 观察vim ~/.vimrc

3. 安装和配置一个插件: ctrlp.vim.

用 `mkdir -p ~/.vim/pack/vendor/start` 创建插件文件夹

下载这个插件: `cd ~/.vim/pack/vendor/start; git clone https://github.com/ctrlpvim/ctrlp.vim`

下载后需要在 ~/.vimrc 中添加如下设置, 参考这里

`set runtimepath\^=~/.vim/pack/vendor/start/ctrlp.vim`

```
ooc@islauc-vm:~/Desktop/tmp/html_root/html/html_root/html$ mkdir -p ~/.vim/pack/vendor/star
t
ooc@islauc-vm:~/Desktop/tmp/html_root/html/html_root/html$ cd ~/.vim/pack/vendor/start; git
clone https://github.com/ctrlpvim/ctrlp.vim
Cloning into 'ctrlp.vim'...
fatal: unable to access 'https://github.com/ctrlpvim/ctrlp.vim/': GnuTLS recv error (-110):
The TLS connection was non-properly terminated.
ooc@islauc-vm:~/vim/pack/vendor/start$ set runtimepath\^=~/.vim/pack/vendor/start/ctrlp.vim
```

图12: 下载插件

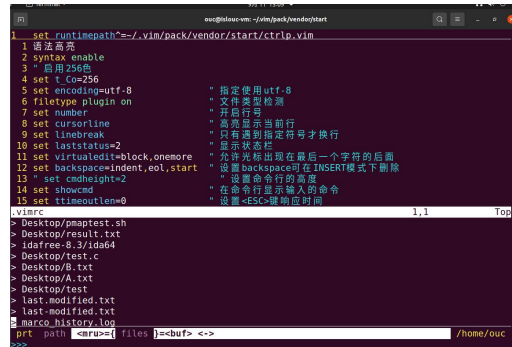


图13：在文件中添加设置

4. 请阅读这个插件的文档。 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。

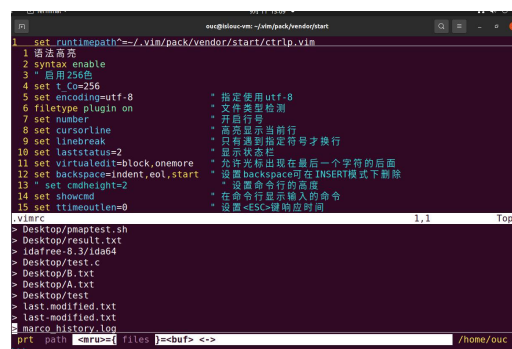


图14：Vim命令控制行：CtrlP

5. 自定义 CtrlP： 添加 configuration 到你的 ~/.vimrc 来用按 Ctrl-P 打开 CtrlP

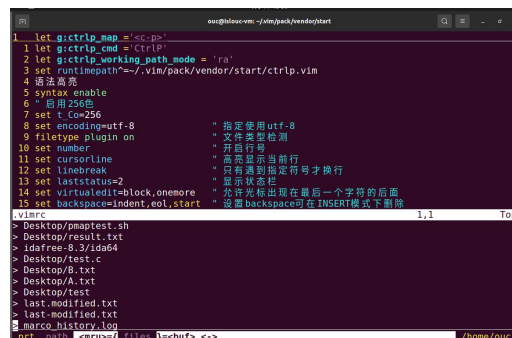


图15：自定义CtrlP

6. 进一步自定义你的 ~/.vimrc 和安装更多插件。 安装插件最简单的方法是使用 Vim 的包管理器，即使用 vim-plug 安装插件：

(1) 首先要安装 vim-plug 插件管理器：

```
curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```

(2) 打开 ~/.vimrc 文件:

```
vim ~/.vimrc
```

在 ~/.vimrc 文件中, 添加以下内容以配置插件管理器和插件:

```
" 初始化 vim-plug
call plug#begin('~/.vim/plugged')
" 安装插件
Plug 'preservim/NERDTree'          " NERDTree 插件
Plug 'wikitopian/hardmode'         " hardmode 插件
" 你可以在这里添加更多插件
" 例如:
" Plug 'junegunn/fzf'
" Plug 'junegunn/fzf.vim'
" 结束插件配置
call plug#end()
```

(3) 保存并退出 ~/.vimrc 文件:

按 Esc 键进入普通模式。

输入 :wq 并按回车保存更改并退出 Vim。

(4) 安装插件

启动 Vim:

```
vim
```

(5) 在 Vim 命令行中运行 :PlugInstall 以安装配置文件中列出的所有插件:

```
:PlugInstall
```

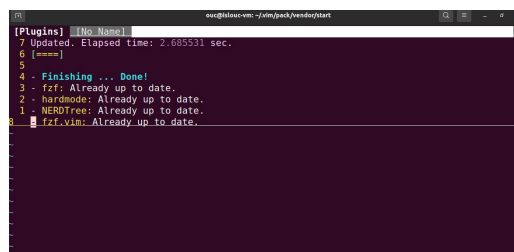


图16: 进一步自定义安装插件

1.3 数据整理

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。

```
cat /usr/share/dict/words | tr "[:upper:]" "[:lower:]" | grep -E
"^[^a]*a){3}.*$" | grep -v "'s$" | wc -l
# 850
```

2. 进行原地替换听上去很有诱惑力, 例如: `sed s/REGEX/SUBSTITUTION/ input.txt > input.txt`。但是这并不是一个明智的做法, 为什么呢? 还是说只有 `sed` 是这样的? 查看 `man sed` 来完成这个问题。

`sed s/REGEX/SUBSTITUTION/ input.txt > input.txt` 表达式中后一个 `input.txt` 会首先被清空, 而且是发生在前的。所以前面一个 `input.txt` 在还没有被 `sed` 处理时已经为空了。在使用正则处理文件前最好是首先备份文件。

```
sed -i.bak s/REGEX/SUBSTITUTION/ input.txt
```

3. 在网上找一个类似 这个 或者这个的数据集。或者从这里找一些。使用 `curl` 获取数据集并提取其中两列数据, 如果您想要获取的是HTML数据, 那么 `pup` 可能会更有帮助。对于JSON类型的数据, 可以试试 `jq`。请使用一条指令来找出其中一列的最大值和最小值, 用另外一条指令计算两列之间差的总和。

```
~$ curl 'https://stats.wikimedia.org/EN/TablesWikipedia22.htm#wikipedians' \
| sed -n "/table/,/</table>/p" \
| grep "ctr" | sed "1,12d"|head -n -3 \
| sed -E 's/(<[>]*>)+/ /g' \
| sed 's/ &nbsp; / /g' \
| sed 's/ &nbsp; / /g' > data

~$ cat data # 处理后的数据为Jan2001截至Oct2018的
Oct2018 2642056 12641 70805 10498 48.9M - 6101 - - - - 10.3M - - - - 42.6M
Sep2018 2629415 11171 66574 10804 48.7M - 6116 - - - - 10.1M - - - - 42.4M
Aug2018 2618244 12058 68688 10640 48.5M - 6839 - - - - 10.2M - - - - 42.1M
Jul2018 2606186 12026 68037 10305 48.3M - 6987 - - - - 9.5M - - - - 41.9M
...
Jan2001 7 7 9 - 31 12 1 8.6 1352 29% 10% 267 301k 3.0k 15 - - 2 163
```

图17网上找 这个

2 解题感悟

通过这次试验, 我了解了shell工具和脚本、编辑器Vim以及数据整理, 发现了`texstudio`运行时的bug: 对插入图片的数量有限制, 插入的多了之后会发现根本无法粘贴进Tex course里面, 对此, 我的解决办法是后期处理实验报告文件, 有些图片是我后期插入进去的。

3 github链接

<https://github.com/zyx-cyber/coursecontent.git>