

实验报告二

姓名: 耿铭骏

September 5, 2024

实验内容概述

本次实验的内容主要是关于 Shell 工具和脚本、编辑器 (Vim) 以及数据整理的练习。通过以下练习, 学习如何使用 Shell 进行命令行操作和脚本编写, 掌握 Vim 编辑器的基本操作和插件配置, 并能够使用命令行工具进行数据的清洗和分析。

练习一: Shell 工具和脚本练习

练习内容

1. 使用 `ls` 命令的多种选项。

解题方法: 使用 `ls -a -h -t --color=auto` 命令显示所有文件 (包括隐藏文件), 以人类可读的格式输出文件大小, 按最近访问顺序排序, 并以彩色文本显示结果。

- `-a`: 显示所有文件, 包括隐藏文件 (以 “.” 开头的文件)。
- `-h`: 以人类可读的格式显示文件大小 (例如, 使用 1K, 234M, 2G 等单位)。
- `-t`: 根据文件的修改时间排序, 最近的文件排在最前面。
- `--color=auto`: 为不同类型的文件显示不同的颜色, 便于区分。

输出:

```
ouc@islouc-vm:~/Desktop$ ls -a -h -t --color=auto
.                test_nx_enabled    experiment4.py
experiment.py.save test_fortify3       experiment3.py
test_full_relro  test_fortify2       experiment2.py
test_partial_relro test_fortify1       experiment.py
test_no_relro    test_default        teast
..               test_stack_protection_all file.txt
test.c           test_stack_protection temp0
test_pie_2       test_no_protection  sqli-labs-mysql.sh
test_pie_1       experiment8.py       Misc
test_no_pie      experiment7.py       Ghidra.desktop
test_nx_enabled_again experiment6.py        'IDA Freeware 8.3.desktop'
test_nx_disabled experiment5.py
ouc@islouc-vm:~/Desktop$
```

2. 编写两个 Bash 函数 marco 和 polo, 实现保存和恢复当前工作目录的功能。

解题方法: 编写以下两个 Bash 函数:

- (a) 打开终端并创建一个新的脚本文件 marco.sh, 使用命令:

```
nano marco.sh
```

- (b) 在文件中编写以下两个 Bash 函数并保存:

```
marco() {  
    export MARCO=$(pwd)  
}  
polo() {  
    cd "$MARCO"  
}
```

- (c) 在终端中运行以下命令, 加载定义的函数:

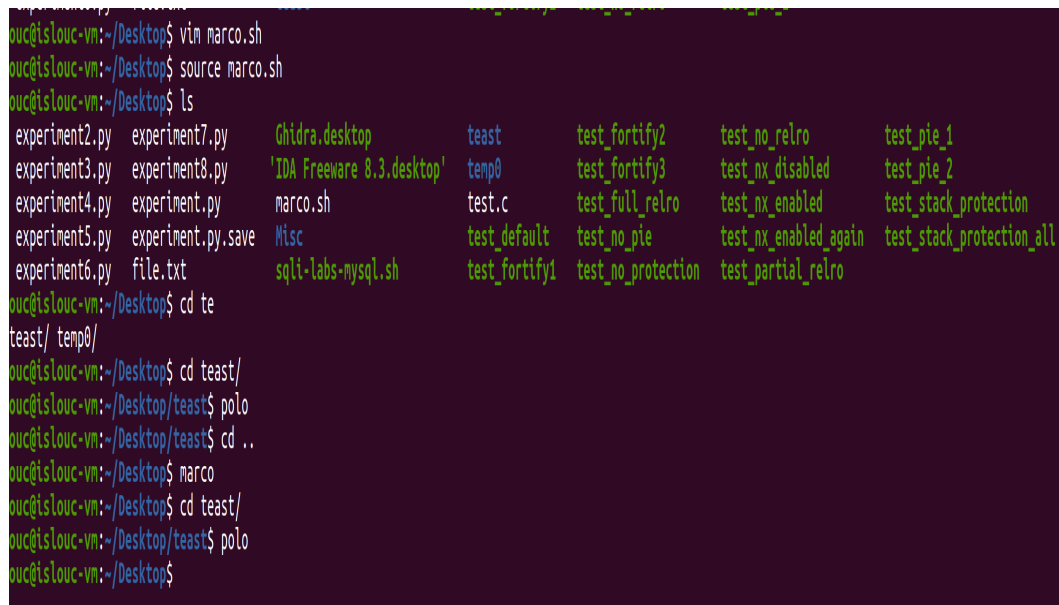
```
source marco.sh
```

- (d) 使用 marco 命令保存当前工作目录:

```
marco
```

- (e) 切换到其他目录, 然后使用 polo 命令恢复到之前保存的目录:

```
polo
```



```
ouc@islouc-vm:~/Desktop$ vi marco.sh  
ouc@islouc-vm:~/Desktop$ source marco.sh  
ouc@islouc-vm:~/Desktop$ ls  
experiment2.py  experiment7.py  Ghidra.desktop  teast            test_fortify2    test_no_relo    test_pie_1  
experiment3.py  experiment8.py  'IDA Freeware 8.3.desktop'  temp0           test_fortify3    test_nx_disabled test_pie_2  
experiment4.py  experiment.py   marco.sh         test.c           test_full_relo   test_nx_enabled  test_stack_protection  
experiment5.py  experiment.py.save Misc             test_default     test_no_pie      test_nx_enabled_again test_stack_protection_all  
experiment6.py  file.txt       sql-labs-mysql.sh test_fortify1    test_no_protection test_partial_relo  
ouc@islouc-vm:~/Desktop$ cd te  
teast/ temp0/  
ouc@islouc-vm:~/Desktop$ cd teast/  
ouc@islouc-vm:~/Desktop/teast$ polo  
ouc@islouc-vm:~/Desktop/teast$ cd ..  
ouc@islouc-vm:~/Desktop$ marco  
ouc@islouc-vm:~/Desktop$ cd teast/  
ouc@islouc-vm:~/Desktop/teast$ polo  
ouc@islouc-vm:~/Desktop$
```

3. 使用 Bash 脚本统计文件夹中的文件数量。

解题方法: 编写如下脚本:

- (a) 打开终端, 创建一个新的脚本文件, 命令如下:

```
nano count_files.sh
```

- (b) 在编辑器中输入以下内容, 编写 Bash 脚本:

```
#!/bin/bash  
count=$(ls -1 | wc -l) # 统计当前文件夹中的文件数量
```

```
echo "Total number of files: $count" # 输出文件数量
```

保存文件并退出编辑器。

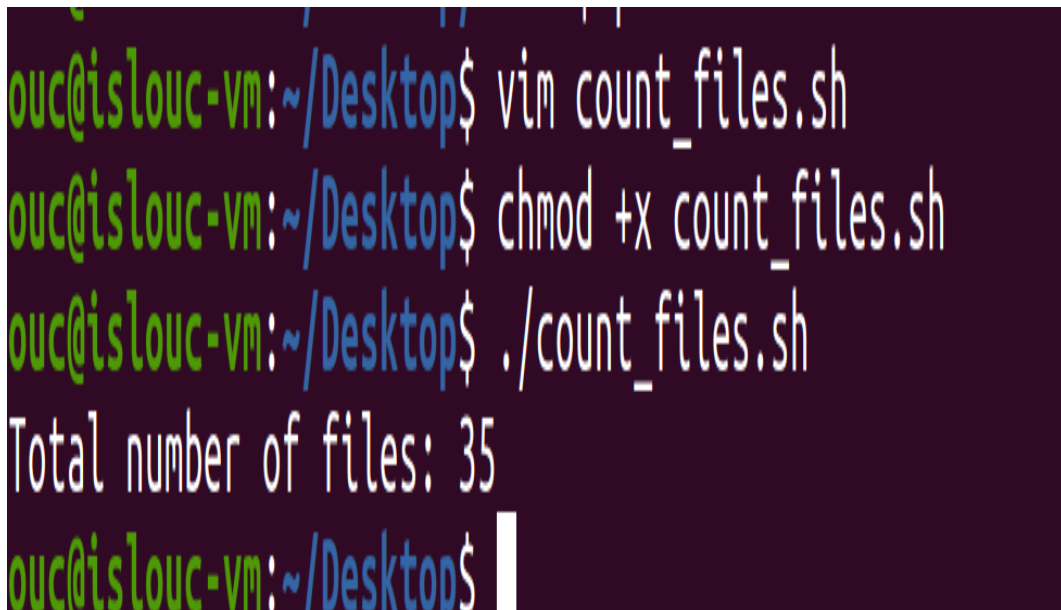
- (c) 使脚本文件具有可执行权限，运行以下命令：

```
chmod +x count_files.sh
```

- (d) 在终端中运行脚本，统计文件夹中的文件数量：

```
./count_files.sh
```

- (e) 观察输出结果，例如：



A terminal window with a dark purple background and light green text. The prompt is 'ouc@islouc-vm:~/Desktop\$'. The user enters 'vim count_files.sh', then 'chmod +x count_files.sh', and finally './count_files.sh'. The output of the script is 'Total number of files: 35'. The prompt returns to 'ouc@islouc-vm:~/Desktop\$'.

4. 调试和重试脚本，运行另一个脚本直到出错并记录输出。

解题方法：编写如下 Bash 脚本，运行另一个脚本 `buggy.sh` 直到它出错，记录所有标准输出和错误流，并在出错时输出失败次数。

- (a) 打开终端，创建一个新的脚本文件，命令如下：

```
nano debug.sh
```

- (b) 在编辑器中输入以下内容，编写 Bash 脚本：

```
#!/usr/bin/env bash
count=0
echo > out.log # 清空日志文件

while true
do
    ./buggy.sh &>> out.log # 运行 buggy.sh 并将所有输出追加到 out.log
    if [[ $? -ne 0 ]]; then # 检查上一个命令的退出状态
        cat out.log # 输出日志文件内容
        echo "failed after $count times" # 输出失败次数
        break # 退出循环
    fi
```

```
((count++)) # 计数器递增
done
```

保存文件并退出编辑器。

- (c) 使脚本文件具有可执行权限，运行以下命令：
`chmod +x debug.sh`
- (d) 在终端中运行脚本，开始调试和重试：
`./debug.sh`
- (e) 脚本将反复运行 `buggy.sh`，直到其出错，并在出错时输出日志和失败次数，例如：
`failed after 34 times`

```
ouc@islouc-vm:~/Desktop$ ./debug.sh  
./debug.sh: line 6: ./buggy.sh: No such file or directory  
failed after 0 times  
ouc@islouc-vm:~/Desktop$ vim buggy.sh  
ouc@islouc-vm:~/Desktop$ ./debug.sh  
./debug.sh: line 6: ./buggy.sh: Permission denied  
failed after 0 times  
ouc@islouc-vm:~/Desktop$ chmod +x buggy.sh  
ouc@islouc-vm:~/Desktop$ ./debug.sh  
  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Everything went according to plan  
Something went wrong  
The error was using magic numbers  
failed after 19 times
```

5. 使用 `find` 和 `xargs` 命令递归查找文件夹中的 HTML 文件并将其压缩成 ZIP 文件。

解题方法：使用如下命令：

- (a) 打开终端，进入要查找 HTML 文件的目标目录：
`cd /demodir`
- (b) 使用 `find` 命令查找所有 HTML 文件，并将其路径传递给 `xargs`，同时使用 `tar` 命令压缩这些文件为一个 ZIP 文件，运行以下命令：
`find . -type f -name "*.html" | xargs -d '\n' tar -cvzf html.zip`
- (c) 等待命令执行完成，查看压缩后的 ZIP 文件 `html.zip` 是否已生成。可以运行以下命令确认：
`ls -lh html.zip`
- (d) 如果需要查看压缩包中的文件，可以使用以下命令：
`tar -tf html.zip`

```

ouc@islouc-vm:~/Desktop$ mkdir demodir
ouc@islouc-vm:~/Desktop$ cd demodir/
ouc@islouc-vm:~/Desktop/demodir$ vim 1.html
ouc@islouc-vm:~/Desktop/demodir$ vim 2.html
ouc@islouc-vm:~/Desktop/demodir$ find . -type f -name "*.html" | xargs -d '\n' tar -cvzf html.zip
./1.html
./2.html
ouc@islouc-vm:~/Desktop/demodir$ ls -lh html.zip
-rw-rw-r-- 1 ouc ouc 126 9月  5 20:32 html.zip
ouc@islouc-vm:~/Desktop/demodir$ tar -tf html.zip
./1.html
./2.html
ouc@islouc-vm:~/Desktop/demodir$

```

6. 使用 `sed` 命令在文件中进行替换。

解题方法： 使用如下命令：

```
sed -i 's/old_text/new_text/g' file.txt
```

```

ouc@islouc-vm:~/Desktop/demodir$ echo "This is old_text. Replace old_text with new_text." > example.txt
ouc@islouc-vm:~/Desktop/demodir$ cat example.txt
his is old_text. Replace old_text with new_text.
ouc@islouc-vm:~/Desktop/demodir$ sed -i 's/old_text/new_text/g' example.txt
ouc@islouc-vm:~/Desktop/demodir$ cat example.txt
his is new_text. Replace new_text with new_text.
ouc@islouc-vm:~/Desktop/demodir$

```

7. 使用 `grep` 和 `wc` 统计包含特定关键字的行数。

解题方法： 使用如下命令：

- (a) 创建一个示例文件 `input.txt` 并添加一些示例文本，运行以下命令：

```
echo -e "apple\nbanana\napple pie\norange\napple tart" > input.txt
```

- (b) 查看文件内容，确认其中包含待搜索的关键字 `apple`：

```
cat input.txt
```

- (c) 使用 `grep` 命令查找包含关键字 `apple` 的行，并使用 `wc` 统计行数，运行以下命令：
- ```
grep "apple" input.txt | wc -l
```
- (d) 查看输出结果，例如：
- ```
3
```
- 表示文件中共有 3 行包含关键字 `apple`。

```
ouc@islouc-vm:~/Desktop/demodir$ echo -e "apple\nbanana\napple pie\norange\napple tart" > input.txt
ouc@islouc-vm:~/Desktop/demodir$ cat input.txt
apple
banana
apple pie
orange
apple tart
ouc@islouc-vm:~/Desktop/demodir$ grep "apple" input.txt | wc -l
3
```

8. 使用 `tr` 命令将文件中的所有字母转换为大写。

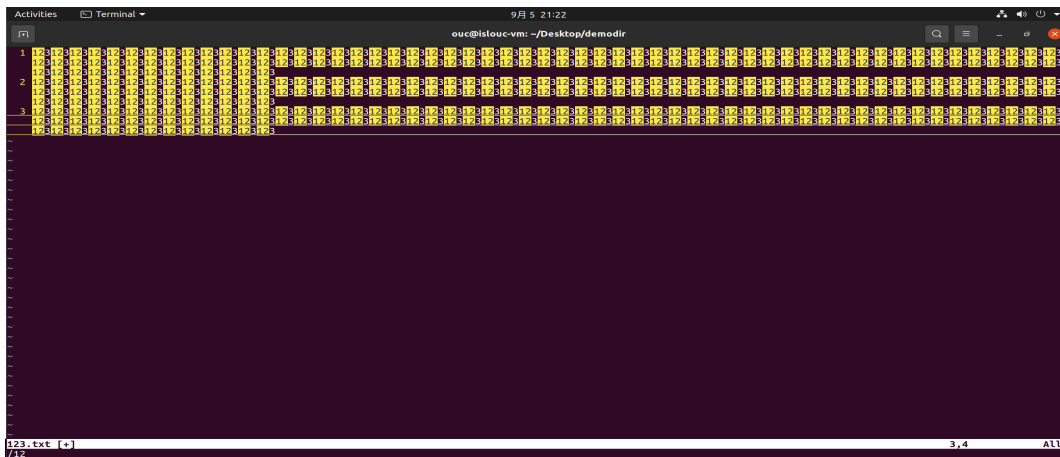
解题方法：使用如下命令：

```
tr '[:lower:]' '[:upper:]' < input.txt > output.txt
```

```
ouc@islouc-vm:~/Desktop/demodir$ echo -e "hello world\nthis is a test file\nconvert me to uppercase" > input.txt
ouc@islouc-vm:~/Desktop/demodir$ cat input.txt
hello world
this is a test file
convert me to uppercase
ouc@islouc-vm:~/Desktop/demodir$ tr '[:lower:]' '[:upper:]' < input.txt > output.txt
ouc@islouc-vm:~/Desktop/demodir$ cat input.txt
hello world
this is a test file
convert me to uppercase
ouc@islouc-vm:~/Desktop/demodir$ cat output.txt
HELLO WORLD
THIS IS A TEST FILE
CONVERT ME TO UPPERCASE
ouc@islouc-vm:~/Desktop/demodir$
```

解题感悟

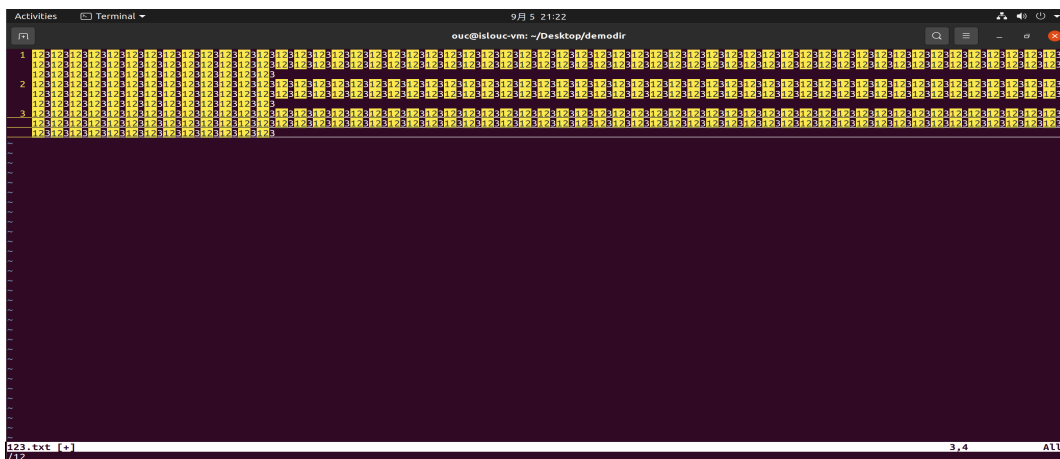
通过这些练习，进一步加深了对 Shell 命令和脚本的理解，Shell 脚本的强大和灵活性在于其能够处理各种复杂的操作，可以提高工作效率。



5. 使用 Vim 的查找和替换功能。

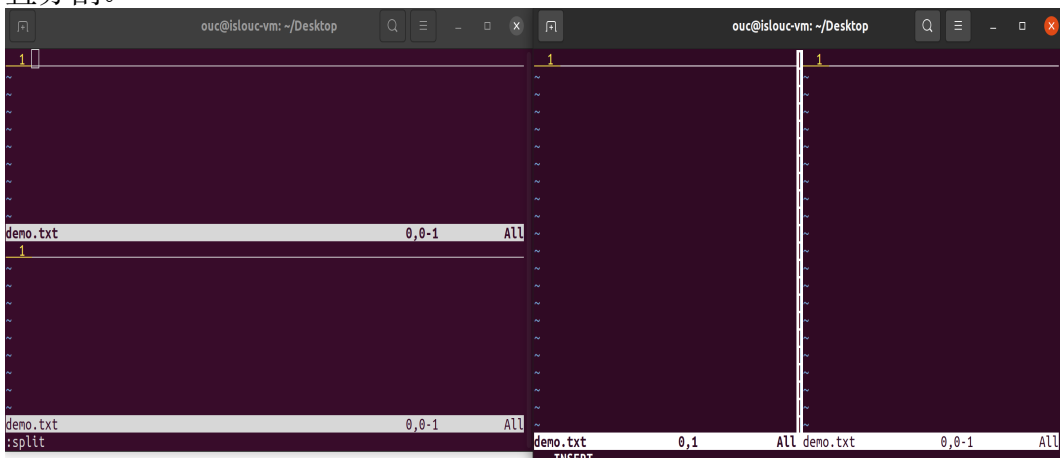
解题方法：在 Vim 中输入如下命令：

```
:%s/old_text/new_text/g
```



6. 学习 Vim 的窗口分割功能。

解题方法：使用 `:split` 或 `:vsplit` 命令在 Vim 中分割窗口，分别进行水平和垂直分割。



7. 自动缩进代码。

解题方法：在 Vim 中打开一个文件，输入 `gg=G` 自动调整整个文件的缩进格式。

解题感悟

Vim 编辑器虽然上手不易，但一旦熟悉其操作和快捷键，编辑效率大大提高。尤其是利用插件和宏功能，可以轻松完成一些复杂的文本编辑任务。

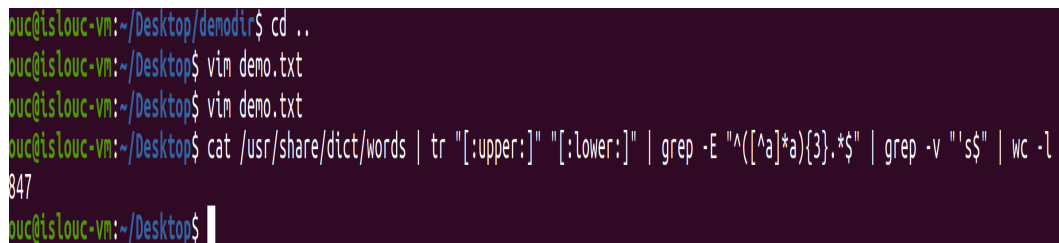
练习三：数据整理练习

练习内容

1. 使用正则表达式统计符合特定条件的单词数量。

解题方法：使用正则表达式和命令行工具组合，统计 words 文件中包含至少三个'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
```

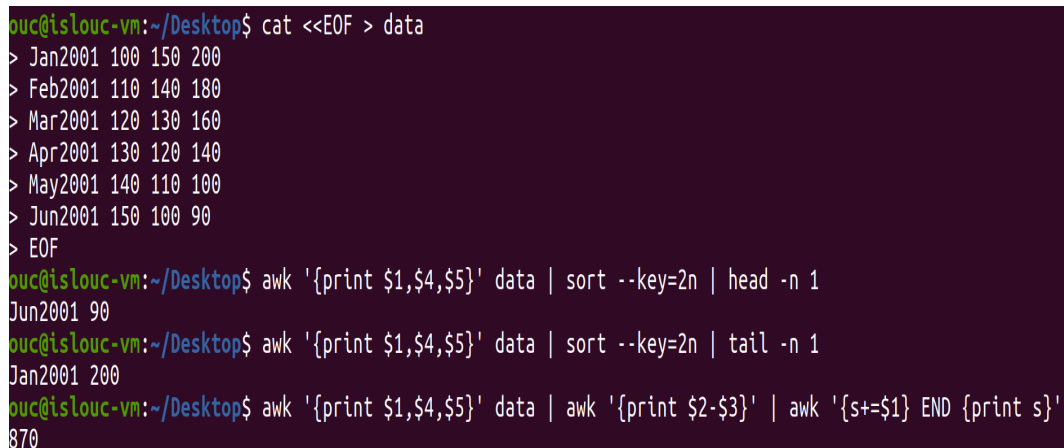


```
buc@islouc-vm:~/Desktop/demodir$ cd ..  
buc@islouc-vm:~/Desktop$ vim demo.txt  
buc@islouc-vm:~/Desktop$ vim demo.txt  
buc@islouc-vm:~/Desktop$ cat /usr/share/dict/words | tr "[:upper:]" "[:lower:]" | grep -E "^(^[a]*a){3}.*$" | grep -v "'s$" | wc -l  
847  
buc@islouc-vm:~/Desktop$
```

2. 查找和分析数据集中列的最大值和最小值。

解题方法：使用 awk 命令从数据集中提取列并计算最大值、最小值及列差的总和。

```
awk '{print $1,$4,$5}' data | sort --key=2n | head -n 1  
awk '{print $1,$4,$5}' data | sort --key=2n | tail -n 1  
awk '{print $1,$4,$5}' data | awk '{print $2-$3}' |  
awk '{s+=$1} END {print s}'
```

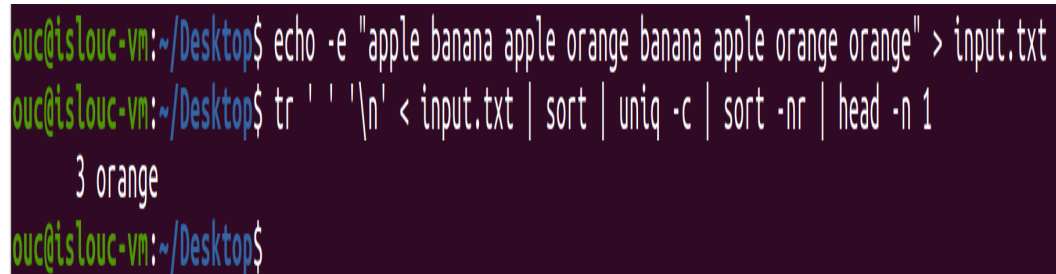


```
buc@islouc-vm:~/Desktop$ cat <<EOF > data  
> Jan2001 100 150 200  
> Feb2001 110 140 180  
> Mar2001 120 130 160  
> Apr2001 130 120 140  
> May2001 140 110 100  
> Jun2001 150 100 90  
> EOF  
buc@islouc-vm:~/Desktop$ awk '{print $1,$4,$5}' data | sort --key=2n | head -n 1  
Jun2001 90  
buc@islouc-vm:~/Desktop$ awk '{print $1,$4,$5}' data | sort --key=2n | tail -n 1  
Jan2001 200  
buc@islouc-vm:~/Desktop$ awk '{print $1,$4,$5}' data | awk '{print $2-$3}' | awk '{s+=$1} END {print s}'  
870
```

3. 使用 `sort` 和 `uniq` 找出文件中出现频率最高的单词。

解题方法：使用如下命令：

```
tr ' ' '\n' < input.txt | sort | uniq -c | sort -nr | head -n 1
```

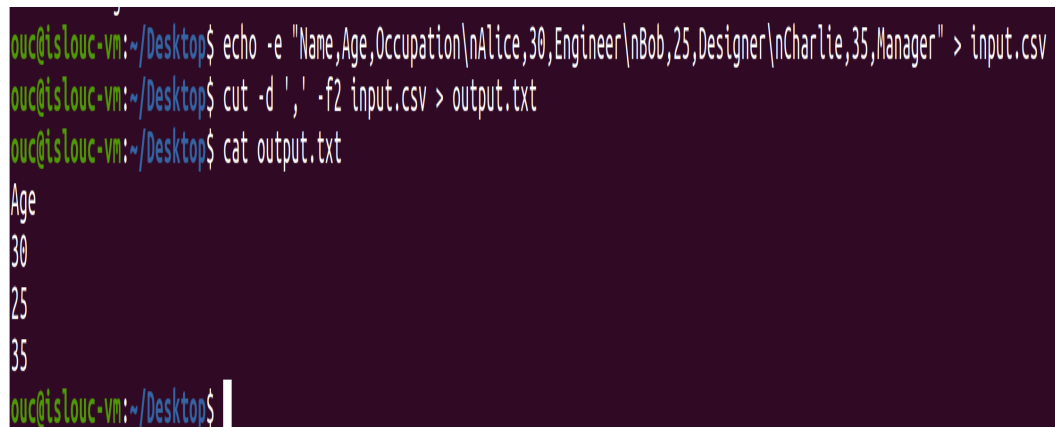


```
ouc@islouc-vm:~/Desktop$ echo -e "apple banana apple orange banana apple orange orange" > input.txt
ouc@islouc-vm:~/Desktop$ tr ' ' '\n' < input.txt | sort | uniq -c | sort -nr | head -n 1
3 orange
ouc@islouc-vm:~/Desktop$
```

4. 使用 `cut` 命令提取 CSV 文件中的特定列。

解题方法：使用如下命令：

```
cut -d ',' -f2 input.csv > output.txt
```



```
ouc@islouc-vm:~/Desktop$ echo -e "Name,Age,Occupation\nAlice,30,Engineer\nBob,25,Designer\nCharlie,35,Manager" > input.csv
ouc@islouc-vm:~/Desktop$ cut -d ',' -f2 input.csv > output.txt
ouc@islouc-vm:~/Desktop$ cat output.txt
Age
30
25
35
ouc@islouc-vm:~/Desktop$
```

5. 使用 `paste` 命令将多个文件的内容合并为一个文件。

解题方法：使用如下命令：

```
paste file1.txt file2.txt > output.txt
```

```
ouc@islouc-vm:~/Desktop$ echo -e "Alice\nBob\nCharlie" > file1.txt
ouc@islouc-vm:~/Desktop$ echo -e "Engineer\nDesigner\nManager" > file2.txt
ouc@islouc-vm:~/Desktop$ paste file1.txt file2.txt > output.txt
ouc@islouc-vm:~/Desktop$ paste file1.txt file2.txt > output.txt
ouc@islouc-vm:~/Desktop$ cat output.txt
Alice  Engineer
Bob    Designer
Charlie Manager
ouc@islouc-vm:~/Desktop$
```

解题感悟

通过这些数据整理的练习，学会了如何高效地使用命令行工具来处理和分析数据。正则表达式的应用和对数据的筛选与统计，使得处理大数据文件变得更加简洁和快速。

GitHub 链接

本次实验报告的源代码已上传到 GitHub，您可以通过以下链接查看完整的报告和代码：
https://github.com/apellidole/vim_shell.git