

实验报告

——用 Shell 工具和脚本与 Vim
编辑器来进行数据整理

姓名：翟一航

学号：23020011046

班级：23级软件工程五八班

1 实验要求

- 1.1 学会使用 Shell 工具来进行脚本编写和运行
- 1.2 学会利用 Vim 编辑器来进行更高效的文本编辑和数据整理
- 1.3 完成至少 4 个课堂练习与 20 个与 Shell 和 Vim 有关的实例

2 实验内容

2.1 Shell 的学习

- 2.1.1 Shell 是一个命令行界面和脚本语言，用于与操作系统交互。它充当用户和系统之间的接口，解释和执行命令，并允许编写脚本以自动化任务。常见的 Shell 包括 Bash、sh、zsh、csh 和 ksh，它们各自提供不同的功能和特性，以满足各种用户需求。
- 2.1.2 Shell 主要具有以下功能：
 - 1. 命令解释器：Shell 解释并执行用户输入的命令。这些命令可以是系统内置的，也可以是外部程序或脚本。
 - 2. 脚本语言：Shell 提供了一种编程环境，允许用户编写脚本来自动化各种任务，如文件操作、系统管理、批处理等。
 - 3. 用户界面：Shell 可以提供命令行界面（CLI），通过它，用户可以直接输入命令和参数来控制计算机系统。
- 2.1.3 常见的 Shell 类型包括：
 - 1.Bash：这是最常用的 Linux 和 macOS 的默认 Shell，兼容 BourneShell (sh)，并加入了许多增强特性。在本次实验中，我主要使用 Git (bush) 来进行 Shell 指令和运用的学习。
 - 2.sh：这是最早的 Unix Shell，由 Steve Bourne 开发。
 - 3.zsh：提供了比 Bash 更强大的功能和更多的扩展，通常用于高级用户和开发者。
 - 4.csh：它的语法与 C 语言类似，主要用于一些特定的 Unix 系统。
 - 5.ksh：由 David Korn 开发，融合了 sh 和 csh 的特性，并提供了额外的功能。

2.2 Vim 编辑器的学习

- 2.2.1 Vim 是一款高度可定制的文本编辑器，基于早期的 Vi 编辑器，提供了许多增强功能。它主要用于编写和编辑源代码，但也可以用于一般的文本编辑。
- 2.2.2 Vim 拥有如下特点：
 - 1. 模式编辑：Vim 采用模式化编辑方式，主要有三种模式：普通模式（用于导航和操作文本）、插入模式（用于输入文本）和命令模式（用于执行命令）。这种模式化设计使得编辑过程更高效。
 - 2. 强大的快捷键和命令：Vim 提供了丰富的快捷键和命令，支持复杂的文本操作和导航，提高编辑效率。
 - 3. 可定制性：用户可以通过配置文件（如.vimrc）定制 Vim 的行为和外观，安装插件以扩展其功能。

2.3.2 使用 Vim 进行数据整理

Vim 是一个强大的文本编辑器，适合于手动编辑和处理文本数据。通过搜索与替换、模式匹配和删除、分割和合并文件、宏录制、列操作等功能可以进行数据整理。

2.3.3 结合使用 awk 和 Viim

在将二者结合使用的过程中，一般先使用 awk 处理和提取数据，生成初步整理的结果文件，再在 Vim 中进一步编辑与格式化数据，以满足特定需求或进行复杂的手动调整。Vim 与 awk 的结合使用，可以使数据整理工作更加高效和精准。

2.4 课堂练习

2.4.1 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. 文件以最近访问顺序排序
4. 以彩色文本显示输出结果

打开 Git(Bush)，在初始位置即可进行该课堂练习的操作。尝试后发现，在 Git(Bush) 中并没有 man 这一功能，也没有 fd 这一功能，不过存在 find 功能，在此就不在查看 ls 的具体功能，直接开始课堂练习。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ man fd
bash: man: command not found

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ man find
bash: man: command not found

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ fd
bash: fd: command not found

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ find
.
./buggy.sh
./marco
./num3
./oout.log
./out.log
./polo
```

1. 如果要查看所有的文件，包括隐藏文件，我们可以使用命令 `ls -a`(或者 `ls -l`，不过该命令为显示所有文件以及其详细信息)，结果如下：

```
LENOVO@DESKTOP-00A7G4V MINGW64 ~  
$ ls -a  
./  
../  
.ld9virtualBox/  
.android/  
.bash_history  
.config/  
.crossnote/  
.dotnet/  
.gitconfig  
.idlerc/  
.lessht  
.matplotlib/  
.obsutilAppData/  
.obsutil_log/  
.obsutilconfig  
.templateengine/  
.texlive2024/  
.vscode/  
'3D Objects'/  
AppData/  
'Application Data'@  
Contacts/  
Cookies@  
Documents/  
Downloads/  
Favorites/  
Links/  
'Local Settings'@  
Music/  
'My Documents'@  
NTUSER.DAT  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
1.regtrans-ms  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
2.regtrans-ms  
NetHood@  
OneDrive/  
PrintHood@  
Recent@  
'Saved Games'/  
Searches/  
SendTo@  
Templates@  
Videos/  
WindowsPcManager/  
ansel/  
jpg  
missing-semester-cn.github.io/  
ntuser.dat.LOG1  
ntuser.dat.LOG2
```

(该图片为 ls -a 的结果)

```

LENOVO@DESKTOP-00A7G4V MINGW64 ~
$ ls -l
total 18254
drwxr-xr-x 1 LENOVO 197121      0 Sep 23 2023 '3D Objects'/
drwxr-xr-x 1 LENOVO 197121      0 Jun 24 19:12 AppData/
lrwxrwxrwx 1 LENOVO 197121     31 Oct 26 2023 'Application Data' -> /c/Users/
LENOVO/AppData/Roaming/
drwxr-xr-x 1 LENOVO 197121      0 Oct 26 2023 Contacts/
lrwxrwxrwx 1 LENOVO 197121     59 Oct 26 2023 Cookies -> /c/Users/LENOVO/App
Data/Local/Microsoft/windows/INetCookies/
drwxr-xr-x 1 LENOVO 197121      0 Oct 26 2023 Documents/
drwxr-xr-x 1 LENOVO 197121      0 Aug 25 19:03 Downloads/
drwxr-xr-x 1 LENOVO 197121      0 Oct 26 2023 Favorites/
drwxr-xr-x 1 LENOVO 197121      0 Oct 26 2023 Links/
lrwxrwxrwx 1 LENOVO 197121     29 Oct 26 2023 'Local Settings' -> /c/Users/LE
NOVO/AppData/Local/
drwxr-xr-x 1 LENOVO 197121      0 Oct 26 2023 Music/
lrwxrwxrwx 1 LENOVO 197121     25 Oct 26 2023 'My Documents' -> /c/Users/LENO
VO/Documents/
-rw-r--r-- 1 LENOVO 197121 11796480 Aug 29 22:41 NTUSER.DAT
-rw-r--r-- 1 LENOVO 197121   65536 Oct 26 2023 NTUSER.DAT{a2332f18-cdbf-11ec-
8680-002248483d79}.TM.blf
-rw-r--r-- 1 LENOVO 197121   524288 Oct 26 2023 NTUSER.DAT{a2332f18-cdbf-11ec-
8680-002248483d79}.TMContainer00000000000000000001.regtrans-ms
-rw-r--r-- 1 LENOVO 197121   524288 Oct 26 2023 NTUSER.DAT{a2332f18-cdbf-11ec-
8680-002248483d79}.TMContainer00000000000000000002.regtrans-ms
lrwxrwxrwx 1 LENOVO 197121     67 Oct 26 2023 NetHood -> '/c/Users/LENOVO/App
Data/Roaming/Microsoft/windows/Network Shortcuts'/
drwxr-xr-x 1 LENOVO 197121      0 Aug 30 08:45 OneDrive/
lrwxrwxrwx 1 LENOVO 197121     67 Oct 26 2023 PrintHood -> '/c/Users/LENOVO/
AppData/Roaming/Microsoft/windows/Printer Shortcuts'/
lrwxrwxrwx 1 LENOVO 197121     56 Oct 26 2023 Recent -> /c/Users/LENOVO/AppD
ata/Roaming/Microsoft/windows/Recent/
drwxr-xr-x 1 LENOVO 197121      0 Oct 26 2023 'Saved Games'/
drwxr-xr-x 1 LENOVO 197121      0 Jun 27 17:07 Searches/
lrwxrwxrwx 1 LENOVO 197121     56 Oct 26 2023 SendTo -> /c/Users/LENOVO/AppD
ata/Roaming/Microsoft/windows/SendTo/
lrwxrwxrwx 1 LENOVO 197121     59 Oct 26 2023 Templates -> /c/Users/LENOVO/A
ppData/Roaming/Microsoft/windows/Templates/
drwxr-xr-x 1 LENOVO 197121      0 Aug 30 08:45 Videos/
drwxr-xr-x 1 LENOVO 197121      0 Jul 30 2023 WindowsPcManager/
drwxr-xr-x 1 LENOVO 197121      0 Oct 23 2023 ansel/
-rw-r--r-- 1 LENOVO 197121      4 Nov 30 2023 jmg
drwxr-xr-x 1 LENOVO 197121      0 Aug 27 16:03 missing-semester-cn.github.io/
-rw-r--r-- 1 LENOVO 197121 3003392 Oct 26 2023 ntuser.dat.LOG1
-rw-r--r-- 1 LENOVO 197121 2752512 Oct 26 2023 ntuser.dat.LOG2
-rw-r--r-- 1 LENOVO 197121     20 Oct 26 2023 ntuser.ini
drwxr-xr-x 1 LENOVO 197121      0 Aug 23 10:10 repos/
drwxr-xr-x 1 LENOVO 197121      0 Sep 4 2023 sangfor/
drwxr-xr-x 1 LENOVO 197121      0 Sep 18 2023 source/
drwxr-xr-x 1 LENOVO 197121      0 Nov 2 2023 wc/
lrwxrwxrwx 1 LENOVO 197121     60 Oct 26 2023 「开始」菜单 -> '/c/Users/LENO
VO/AppData/Roaming/Microsoft/windows/Start Menu'/

```

(该图片为 ls -l 的结果, 可以看出所展示的信息要更为详细)

2. 显示文件, 并将文件打印以人类可以理解的格式输出, 在此我们需要输入指令 ls -h。结果如下图。

```
LENOVO@DESKTOP-00A7G4V MINGW04 ~  
$ ls -h  
'3D Objects' /  
AppData /  
'Application Data' @  
Contacts /  
Cookies @  
Documents /  
Downloads /  
Favorites /  
Links /  
'Local Settings' @  
Music /  
'My Documents' @  
NTUSER.DAT  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
1.regtrans-ms  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
2.regtrans-ms  
NetHood @  
OneDrive /  
PrintHood @  
Recent @  
'Saved Games' /  
Searches /  
SendTo @  
Templates @  
Videos /  
windowsPcManager /  
ansel /  
jmg  
missing-semester-cn.github.io /  
ntuser.dat.LOG1  
ntuser.dat.LOG2  
ntuser.ini  
repos /  
sangfor /  
source /  
wc /  
「开始」菜单 @
```

3. 将文件以最近访问顺序排序，用到指令 `ls -h`。结果如下图。

```
LENOVO@DESKTOP-00A7G4V MINGW64 ~  
$ ls -t  
OneDrive/  
Videos/  
NTUSER.DAT  
missing-semester-cn.github.io/  
Downloads/  
repos/  
Searches/  
AppData/  
jmg  
wc/  
Music/  
Links/  
Saved Games'/  
Contacts/  
Favorites/  
ntuser.ini  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
1.regtrans-ms  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
2.regtrans-ms  
ntuser.dat.LOG1  
ntuser.dat.LOG2  
Cookies@  
Local Settings'@  
Recent@  
SendTo@  
Templates@  
「开始」菜单@  
Application Data'@  
NetHood@  
PrintHood@  
My Documents'@  
Documents/  
anse1/  
3D Objects'/  
source/  
sangfor/  
WindowsPcManager/
```

4. 以彩色文本显示输出结果，正常情况下需要输入指令 `ls -color=auto`，不过由于使用的 Git(Bush) 自带了该功能，故与上述结果有些相似。结果如图。


```
LENOVO@DESKTOP-00A7G4V MINGW64 ~  
$ ls --color=auto  
'3D objects'/  
AppData/  
'Application Data'@  
Contacts/  
Cookies@  
Documents/  
Downloads/  
Favorites/  
Links/  
'Local Settings'@  
Music/  
'My Documents'@  
NTUSER.DAT  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
1.regtrans-ms  
NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer00000000000000000000  
2.regtrans-ms  
NetHood@  
OneDrive/  
PrintHood@  
Recent@  
'Saved Games'/  
Searches/  
SendTo@  
Templates@  
Videos/  
WindowsPcManager/  
ansel/  
jmg  
missing-semester-cn.github.io/  
ntuser.dat.LOG1  
ntuser.dat.LOG2  
ntuser.ini  
repos/  
sangfor/  
source/  
WC/  
「开始」菜单@
```

2.4.2 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

在此处我首先通过 vim 编辑功能，创建了两个脚本文件 marco 和 polo，用来写入需要的代码，如图

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim marco

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim polo
```

然后开始编写 marco 和 polo 这两个函数，函数内容如下图所示：

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

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

- 2.4.3 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。加分项：报告脚本在失败前共运行了多少次。

```
#!/usr/bin/env bash

n=$(( RANDOM % 100 ))

if [[ n -eq 42 ]]; then
    echo "Something went wrong"
    >&2 echo "The error was using magic numbers"
    exit 1
fi

echo "Everything went according to plan"
```

首先，创建一个文件名为 buggy.sh 用来存放上述脚本，然后，创建一个 num3 脚本用来运行 buggy.sh。

由此可知，该 buggy.sh 脚本在运行 39 次后出现错误。

2.4.4 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

首先我们使用 mkdir 创建所需要的文件夹，然后再执行 find 命令 `find html_root -name "*.html" -print0 | xargs -0 tar vcf html.zip`，即可查找文件夹中所有的 HTML 文件，并将它们压缩为 zip 文件。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ mkdir html_root

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ cd html_root

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root
$ touch {1..10}.html

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root
$ mkdir html

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root
$ cd html

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ touch try.html

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ tree
bash: tree: command not found

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ find html_root -name "*.html" -print0 | xargs -0 tar vcf html.zip
find: 'html_root': No such file or directory
tar: Cowardly refusing to create an empty archive
Try 'tar --help' or 'tar --usage' for more information.

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ find . -type f -name "*.html" | xargs -d '\n' tar -cvzf html.zip
./try.html

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ cd /d/test2

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ find html_root -name "*.html" -print0 | xargs -0 tar vcf html.zip
html_root/1.html
html_root/10.html
html_root/2.html
html_root/3.html
html_root/4.html
html_root/5.html
html_root/6.html
html_root/7.html
html_root/8.html
html_root/9.html
html_root/html/try.html
```

2.4.5 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。

输入指令 `find . -type f -print0 | xargs -0 ls -lt | head -1` 即可完成该任务。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ find . -type f -print0 | xargs -0 ls -lt | head -1
-rw-r--r-- 1 LENOVO 197121 10240 Aug 30 09:59 ./html.zip
```

3 实验中遇到的问题与解决方法

3.1 想用 man 指令查看相关语句含义时，出现该指令无效的提示

```
LENOVO@DESKTOP-00A7G4V MINGW64 ~
$ man git
bash: man: command not found
```

要想解决这个问题，首先用 which 命令检查 man 命令的安装情况，显示 no man in……则表明没有安装 man 命令，此时输入命令 sudo apt-get install man-db，将 man 命令安装并放到相应的路径中，即可使用 man 命令查看相关命令的功能。

3.2 在进行课堂练习 4 创建文件并对文件做出相关操作时，执行 find 命令时显示无法找到相关文件

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ find html_root -name "*.html" -print0 | xargs -0 tar vcf html.zip
find: 'html_root': No such file or directory
tar: Cowardly refusing to create an empty archive
Try 'tar --help' or 'tar --usage' for more information.

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ find . -type f -name "*.html" | xargs -d '\n' tar -cvzf html.zip
./try.html

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2/html_root/html
$ cd /d/test2
```

这是因为创建文件后在文件夹的最小 html 文件中，不是在文件夹的目录中，因而 find 命令找不到具体执行的目标，应该返回到文件夹所在目录，在进行 find 操作。

4 实例练习

4.1 查看当前所在路径

输入命令 pwd，即可查看当前所在路径。

```
LENOVO@DESKTOP-00A7G4V MINGW64 ~  
$ pwd  
/c/Users/LENOVO
```

4.2 创建脚本文件

输入命令 `touch` 文件名即可创建脚本文件。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ touch marco  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ ls  
marco
```

4.3 运行脚本文件

`./文件名`，即可运行。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ touch num3  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ vim num3  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ num3  
bash: num3: command not found  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ ~$ ./debug.sh  
bash: ~$: command not found  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ ./debug.sh  
bash: ./debug.sh: No such file or directory  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ ./num3  
./num3: line 6: ./buggy.sh: No such file or directory  
failed after 0 times  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ vim num3  
  
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2  
$ vim buggy.sh
```

4.4 打印

创建脚本文件，并进入编辑模式，利用 echo 语句即可打印相关的内容

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim helloworld

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./helloworld
Hello world!
```

```
echo "Hello world!"
```

4.5 统计当前目录下的文件数目

编写脚本（如图），然后运行即可

```
count=$(ls -l | wc -l)
echo "There are $count files in the current directory."
try wc --help for more information.
There are 7 files in the current directory.
```

4.6 按时间戳备份文件

编写脚本（如图），然后运行即可

```
filename="t3.txt"
timestamp=$(date +%Y%m%d%H%M%S)
cp "$filename" "${filename}_${timestamp}_backup"
echo "Successfully"

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./t3
Successfully
```

4.7 在当前目录及其子目录中删除空的子目录。

首先在当前目录中创建空文件夹 t4，然后编写脚本并运行（如图）

html_root	2024/8/30 9:54	文件夹	
t4	2024/9/3 17:16	文件夹	
buggy.sh	2024/8/30 9:48	sh_auto_file	1 KB
helloworld	2024/9/3 16:59	文件	1 KB
html.zip	2024/8/30 9:59	压缩(zipped)文件夹	10 KB
marco	2024/8/30 9:35	文件	1 KB
num2	2024/9/3 17:07	文件	1 KB
num3	2024/8/30 9:45	文件	1 KB
oout.log	2024/8/30 9:48	文本文档	1 KB
out.log	2024/8/30 9:48	文本文档	2 KB
polo	2024/8/30 9:36	文件	1 KB
t3	2024/9/3 17:13	文件	1 KB
t3.txt	2024/9/3 17:11	文本文档	1 KB
t3.txt_20240903_backup	2024/9/3 17:13	TXT_20240903_BA...	1 KB

```
find . -type d -empty -exec rmdir {} \;
```

```
LENOVO@DESKTOP-OOA7G4V MINGW64 /d/test2
```

```
$ vim t4.sh
```

```
LENOVO@DESKTOP-OOA7G4V MINGW64 /d/test2
```

```
$ ./t4.sh
```

删除后的目录如图所示，可见删除成功

html_root	2024/8/30 9:54	文件夹	
.t4.sh.swp	2024/9/3 17:19	SWP 文件	4 KB
buggy.sh	2024/8/30 9:48	sh_auto_file	1 KB
helloworld	2024/9/3 16:59	文件	1 KB
html.zip	2024/8/30 9:59	压缩(zipped)文件夹	10 KB
marco	2024/8/30 9:35	文件	1 KB
num2	2024/9/3 17:07	文件	1 KB
num3	2024/8/30 9:45	文件	1 KB
oout.log	2024/8/30 9:48	文本文档	1 KB
out.log	2024/8/30 9:48	文本文档	2 KB
polo	2024/8/30 9:36	文件	1 KB
t3	2024/9/3 17:13	文件	1 KB
t3.txt	2024/9/3 17:11	文本文档	1 KB
t3.txt_20240903_backup	2024/9/3 17:13	TXT_20240903_BA...	1 KB
t4.sh	2024/9/3 17:18	sh_auto_file	1 KB

4.8 进程监控与重启

编写脚本如下

```
process_name="my_server"
if ! pgrep -x "$process_name" > /dev/null; then
    /path/to/start_script.sh &
fi
```

4.9 定时执行任务

创建一个每分钟执行的任务，并执行。

```
echo "* * * * * /path/to/script.sh" >> ~/.crontab
crontab ~/.crontab

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./t6
* * * * * /path/to/script.sh
```

4.10 文件内容替换

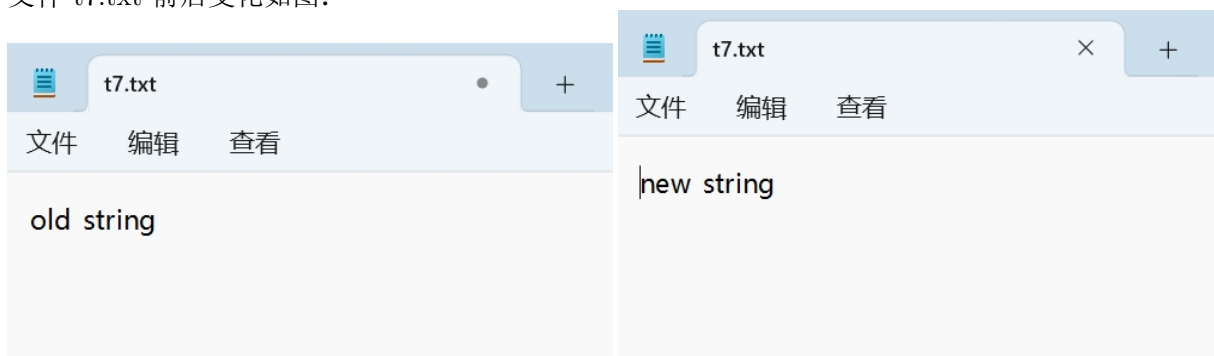
替换文本文件中的特定字符串：用 new_string 替换 old_string，并将结果保存到 t7.txt 中。

```
sed -i 's/old_string/new_string/g' t7.txt

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim t7

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./t7
```

文件 t7.txt 前后变化如图：



4.11 批量重命名文件



将目录中的 txt 文件全部修改为 md 文件。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim t8

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./t8

for file in *.txt; do
    mv "$file" "${file%.txt}.md"
done
```

脚本运行前后 txt 文件变化：

 t7.txt	2024/9/3 17:32	文本文档	1 KB
 t7.md	2024/9/3 17:32	Markdown 源文件	1 KB

4.12 查找并删除指定名称的文件

查找并删除当前目录及其子目录下指定名称的文件。此处删除 t7.md 文件为例，由于结果显著且简单，此处不再展示图片。

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim t9

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./t9

filename="t7.md"
find . -name "${filename}" -delete
```

4.13 检查网络连接

测试网络连通性

```
ping -c 1 google.com && echo "网络连接正常" || echo "网络连接异常"
~
~

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim t10

LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ ./t10
Access denied. Option -c requires administrative privileges.
网络连接异常
```

4.14 打开并保存文件

打开文件: `vim filename.txt`

保存文件: :w 或:wq (保存并退出)

```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim t10
```

```
t10 [unix] (17:46 03/09/2024)
:wq
```

4.15 文本编辑

[illegible]

4.16 移动光标

```
for file in *.txt; do
    mv "$file" "${file%.txt}.md"
done
~
~
~
~
~
~
~
~
~
~
t8 [unix] (17:37 03/09/2024)
```

w: 移动到下一个单词的开头

```
for file in *.txt; do
    mv "$file" "${file%.txt}.md"
done
```

b: 移动到前一个单词的开头

```
for file in *.txt; do
    mv "$file" "${file%.txt}.md"
done
```

gg: 跳转到文件的第一行

```
|for file in *.txt; do
    mv "$file" "${file%.txt}.md"
done
```

G: 跳转到文件的最后一行

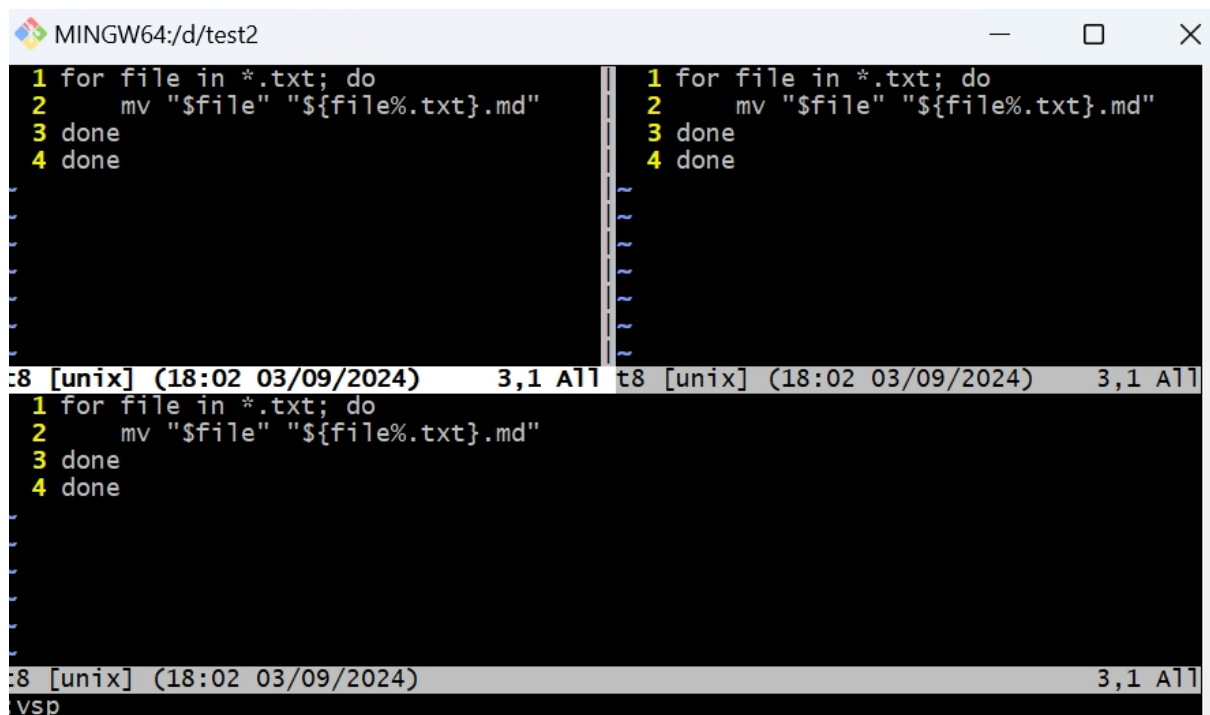
```
for file in *.txt; do
    mv "$file" "${file%.txt}.md"
|done
```

4.17 复制和粘贴

yy: 复制当前行 P (大写): 粘贴到光标后


```
1 for file in *.txt; do
2     mv "$file" "${file%.txt}.md"
3 done
4 done
~
~
~
~
~
~
~
t8 [unix] (18:02 03/09/2024)
1 for file in *.txt; do
2     mv "$file" "${file%.txt}.md"
3 done
4 done
~
~
~
~
~
~
~
t8 [unix] (18:02 03/09/2024)
:split
```

垂直分割窗口：:vsplit 或简写为:vsp



```

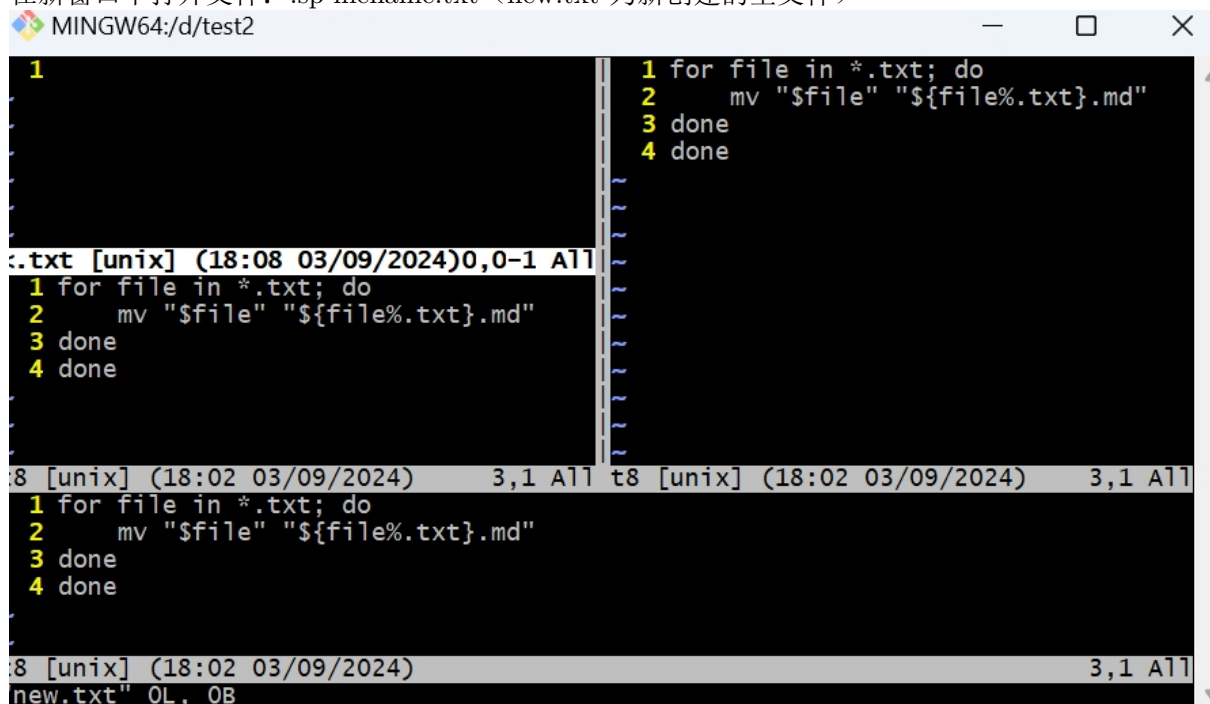
MINGW64:/d/test2
1 for file in *.txt; do
2     mv "$file" "${file%.txt}.md"
3 done
4 done

:8 [unix] (18:02 03/09/2024) 3,1 A11
1 for file in *.txt; do
2     mv "$file" "${file%.txt}.md"
3 done
4 done

:8 [unix] (18:02 03/09/2024) 3,1 A11
vsp

```

在新窗口中打开文件：:sp filename.txt（new.txt 为新创建的空文件）



```

MINGW64:/d/test2
1
2
3
4

:8 [unix] (18:08 03/09/2024) 0,0-1 A11
1 for file in *.txt; do
2     mv "$file" "${file%.txt}.md"
3 done
4 done

:8 [unix] (18:02 03/09/2024) 3,1 A11
1 for file in *.txt; do
2     mv "$file" "${file%.txt}.md"
3 done
4 done

:8 [unix] (18:02 03/09/2024) 3,1 A11
new.txt" 0L, 0B

```

4.20 多文件编辑

使用 vim 打开多个文件：vim file1.txt file2.txt


```
LENOVO@DESKTOP-00A7G4V MINGW64 /d/test2
$ vim old.txt new.txt
2 files to edit
```

在文件间切换: `:next (:n)` 到下一个文件, `:prev (:p)` 到上一个文件

123 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	456 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
old.txt [unix] (18:11 03/09/2024)	new.txt [unix] (18:11 03/09/2024)
"old.txt" [noeol][unix] 1L, 3B	"new.txt" [noeol][unix] 1L, 3B

列出所有打开的文件: `:buffers`

```
:buffers
1 # "old.txt" line 1
2 %a "new.txt" line 1
Press ENTER or type command to continue
```

5 实验收获与感悟

学习 Shell 脚本与工具让我深刻体会到自动化操作的力量，它们极大地提高了处理系统任务和批量文件操作的效率。尽管 Shell 脚本的灵活性和可移植性带来便利，但也需注意不同 Linux 环境间的差异。与高级语言相比，Shell 脚本更侧重于系统级任务的快速实现，展现了其独特的价值。

Vim 编辑器以其高效和灵活著称，通过掌握其多种模式和快捷键，我能够显著提升文本编辑的速度和效率。然而，Vim 的学习曲线较为陡峭，需要时间和实践来适应其独特的操作方式。尽管如此，一旦掌握，Vim 的强大功能和扩展性将为我提供无尽的便利和可能性。

将 Vim 和 awk 结合使用进行数据整理，我感受到了两者在文本处理方面的强大互补性。Vim 提供

了便捷的文本编辑功能，而 `awk` 则擅长复杂的数据分析和处理。这种协同工作的方式不仅提高了数据整理的效率和准确性，还让我在实践中不断发现新的应用场景和解决方案，进一步加深了对这两个工具的理解和掌握。

Github 仓库链接: <https://github.com/Locuslaer/-.git>