实验报告

陈俊泽

August 2024

目录

1 练习实例

Shell 课后题部分

实例 1：为了查看 shell 是否满足要求，使用 echo $SHELL，根据输出可以查看

实例 2: 在 tmp 下新建一个名为 missing 的文件夹：使用 mkdir 命令，具体为 mkdir missing

实例 3：用 man 查看程序 touch 的使用手册: 使用命令 man touch 即可

实例 4: 使用 chmod 命令改变权限，使./semester 能够成功执行; 对于 chmod，有以下规定 4：读权限 (r)2：写权限 (w)1：执行权限 (x)，且使用数字时，权限的顺序为所有者，组用户和其他用户，所以只需要 chmod 777 semester 即可

实例 5：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：首先要将日期信息输出，再使用重定向符号

> 来将信息输入到文件中，具体实现为：stat -c %y' semester>modified.txt

实例 6：写一段命令来从 /sys 中获取笔记本的电量信息，或者台式机 CPU 的温度：cat/sys/class/power_supply/BAT1/capacity

实例 7：find 的用法，find 可以递归的查询符合要求的文件，参数的规定如下：-name （名称为）-path（路径中含有）-mtime（修改时间）-size -size（文件大小范围）-d（文件夹）-f () 文件

实例 8：grep 的用法,grep 用于用于对输入文本进行匹配的通用工具，参数的规定如下：-C（获取查找结果的上下文）-v（也就是输出不匹配的结果）“” filename（搜索指定字符串）“string”（忽略大小写）

实例 9: 然后使用 ls 命令进行如下操作：所有文件（包括隐藏文件）文件打印以人类可以理解的格式输出文件以最近访问顺序排序以彩色文本显示输出结果: 命令应为 ls -hAt –color=auto, 其中 -h：以人类可读的格式显示文件大小（例如 454M）。-A：显示所有文件，包括隐藏文件，但不显示。-t：按修改时间排序（最新的文件在最前面）。–color=auto：以彩色文本显示输出结果。

实例 10：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: 代码的思路如下，首先是 marco 函数先用 touch 在主目录建立 txt 文件，再利用重定向将当前文档路径输入到这个 txt 文件中。polo 函数的思路为利用 cat 读取这个 txt 文件的内容，再利用 cd 命令进入到此路径即可。

实例 11: 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：代码思路如下先用一个变量记录次数，接着用 while 循环，若脚本正常运行，则输出到 stdout.txt 中，否则就输出到 stderr.txt。while 判断结束的条件为脚本退出是否为 1，若为 1，则输出次数，否则令次数加一即可。

实例 12: 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：find . -name "*.html" | xargs tar czvf test.tar.gz，前半部分是查找当前目录及其子目录中所有扩展名为.html 的文件，后半部分是将 find 命令找到的文件列表传递给 tar 命令，tar 将这些文件打包并压缩成 test.tar.gz 文件。

实例 13: 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 -mmin 600 | ls -lt，前半部分是在当前目录及其子目录中查找所有在过去 600 分钟内修改过的文件，后半部分是将找到的文件传递给 ls -lt，该命令按时间降序排列并显示文件的详细信息。

vim 的运用：
实例 14：vim 是一个编辑器，可以对文件进行编辑，使用方法如下：vim + 文件便可进入页面，然后按下 i 键便可以开始编辑。

实例 15: 编辑完后想要退出，先按下 ESC 键，输入如下命令达到不同效果:q 退出（关闭窗口）:w 保存（写）:wq 保存然后退出

实例 16：vim 还可用于查找文本，如\abc 可以从光标所在位置向前查找字符串 abc，\^abc,\abc$ 查找以 abc 为行尾的行,?abc 从光标所在为主向后查找字符串 abc 查找以 abc 为行首的行.

数据统计实例 17: 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。这些单词中，出现频率前三的末尾两个字母是什么? sed 的 y 命令，或者 tr 程序也许可以帮你解决大小写的问题。共存在多少种词尾两字母组合?：命令为 cat /usr/share/dict/words | tr "[:upper:]" "[:lower:]" | grep -E "\(.*a)3.*$" | grep -v "'s$" | wc -l，其中 cat 是打开文件，tr 命令是将大写变为小写，grep -E 部分是使用正则表达式，并且匹配零个或多个非"a" 字符，后跟一个"a" 字符匹配 3 次，之后匹配任意字符。而 grep -v 部分是匹配以's 结尾的字符，grep -v 不显示匹配字符，最后 wc -l 这个命令计算输出的行数，输出结果为符合要求的单词数。

实例 18：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：原表达式中后一个 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。所以在使用正则处理文件前最好是首先备份文件。正确的应为 sed -i.bak s/REGEX/SUBSTITION/ input.txt。

实例 19: 找出您最近十次开机的开机时间平均数、中位数和最长时间：编写程序，代码如下 #!/bin/bash
for i in {0..9}; do
journalctl -b-$i | grep "Startup finished in"
done
此段代码放在 getlog.sh 中，作用是输出最近十次开机的时间，然后利用重定向输出到 txt 文件，./getlog > starttime.txt，最后便可以找到开机时间平均数、中位数和最长时间。

实例 20：查看之前三次重启启动信息中不同的部分（参见 journalctl 的 -b 选项):cat last3start.txt | sed -E "s/.*pi (.*)^1/" | sort | uniq -c | sort | awk '$1!=3 { print }'，这段命令按顺序的作用为从 last3start.txt 文件中提取每一行 pi 后面的内容，对提取的内容进行排序，统计每个唯一内容的出现次数，再次排序，过滤出出现次数不等于 3 的内容并打印。

2 解题感悟

通过本次实验，我对 Shell 命令、Vim 编辑器以及数据整理有了更加深入的理解。
对于 shell，我通过具体实例学习了 Shell 的基本操作，比如文件和目录的创建（如 mkdir）、权限管理（如 chmod）和文件输出重定向（如 >）。这些操作不仅帮助我熟悉了基本命令的使用，还加深了我对文件系统管理和命令组合的理解。例如，通过使用 find 命令，我能递归查找符合条件的文件，这对于管理大型文件夹结构特别有用。Shell 命令的强大之处在于它的组合能力，例如，通过 | 和 >，可以将多个命令的输出作为输入，实现复杂的任务。
对于 vim，我练习了基本的编辑操作，如插入模式（按 i 键）和保存退出（:wq）等等。Vim 的高效在于其模式化编辑方式，使得文本操作更加灵活。

并且通过掌握基本命令和快捷键，我能更高效地进行文本编辑
对于数据整理, 在实验中，我使用了多种工具和命令来处理数据。通过 grep 和 sed，我能够筛选和替换文本数据，这些工具帮助了我处理大量数据。这些工具能够有助于筛选、替换和统计数据，提升数据整理和分析的效率。