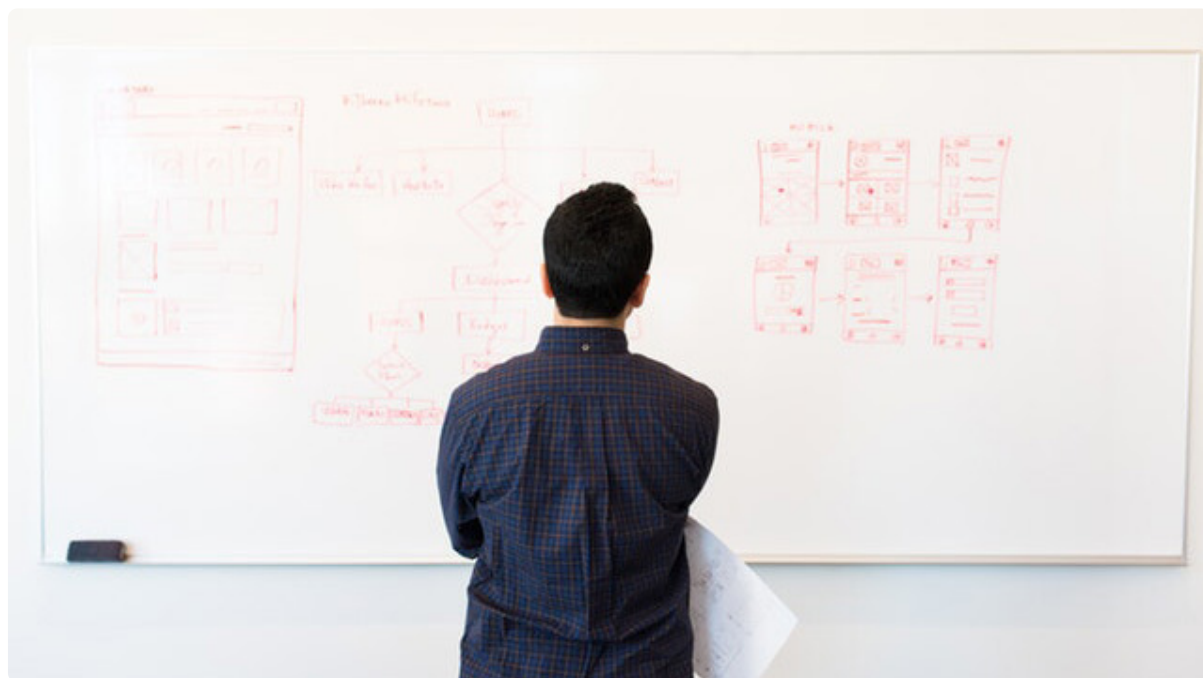


22 数据处理，慢条斯理

更新时间：2019-07-15 11:17:35



“

成功的奥秘在于目标的坚定。

——迪斯雷利

”

内容简介

1. 前言
2. grep 命令：筛选数据
3. sort 命令：为文件排序
4. wc 命令：文件的统计
5. uniq 命令：删除文件中的重复内容
6. cut 命令：剪切文件的一部分内容
7. 总结

1. 前言

上一课是测试题。

终于到了第三部分了，有点激动，如果前两个部分是我们的系列课程的初级篇，那么从第三部分开始，我们就要进入高级篇咯。

我也知道大家学习辛苦了，所以请对自己说：“可喜可贺，掌声给自己！”

好了好了，我重回淡定。大家看到今天的标题应该会对这一课的内容很有兴趣吧，毕竟我们每天都在跟各种数据打交道。

Linux 的文件里也包含各种数据，所以数据处理就显得尤为重要。

之前的课中已经介绍过：大部分 Linux 的命令是基于 Unix 操作系统的模式，当然源码是重写的。

因此，Linux 虽是 1991 年问世的，但是其很多设计理念和命令却沿用了 20 世纪 60 年代的模式。

这样的事实对我们学习 Linux 的人有个好处：不必每隔一段时间就学新东西，很多知识点可以沿用很久。例如现在近 60 岁的一个 Unix 的老工程师，操作 Ubuntu 等新近 Linux 发行版基本没有什么问题。

你也许还是会问这个问题：为什么过了这么多年，好多 Linux 命令都没变呢？

那是因为没有必要变。因为大多数 Linux 命令都具有很基本的功能，而且它们在自己的岗位上敬忠职守，工作做得很棒，这些命令都是 Linux 系统的“基石”。

这一课我们将学习好多个基本的命令，这些命令用于提取、排序、筛选文件中的各种数据，内容也是很轻松的。

这些命令中，有些你以后几乎每天都会用到，例如 grep 命令。

好了，闲话说完，该动手实践了。

2. grep 命令：筛选数据

grep 是 Globally search a Regular Expression and Print 的缩写，意思是“全局搜索一个正则表达式，并且打印”。

意思不太好理解吧？没关系，也不需要深刻理解原意。

grep 命令的功能简单说来是在文件中查找关键字，并且显示关键字所在的行。

grep 命令极为强大，也是 Linux 中使用最多的命令之一。它的强大之处在于它不仅可以实现简单的查找，而且可以配合 [正则表达式](#) 来实现比较复杂的查找。

至于什么是正则表达式，大家有兴趣可以去百度或 Google 来学习一下，这是程序员需要掌握的知识点之一。

正则表达式提供了搜索文本的一种高级方式。我们不仅在 Linux 的命令行中用到正则表达式，而且在很多的文本编辑器里也用到，在许多编程语言例如 C++，Java，PHP 等等也会用到。

首先，我们学习 grep 的简单用法。之后我们再学习如何配合正则表达式来实现复杂的查找。

grep 的简单用法

grep 的使用方法有很多种，我们一开始先学习最基本的用法：

```
grep text file
```

可以看到，上面就是 grep 命令的最基本用法。

text 代表要搜索的文本，file 代表供搜索的文件。

我们用实际的例子来学习：比如我要在用户的家目录的 .bashrc 文件中搜索 alias 这个文本，而且显示所有包含 alias 的行。

```
grep alias .bashrc
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ grep alias .bashrc
# enable color support of ls and also add handy aliases
alias ls='ls --color=auto'
#alias dir='dir --color=auto'
#alias vdir='vdir --color=auto'
alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
# some more ls aliases
alias ll='ls -aF'
alias la='ls -A'
alias l='ls -CF'
# Add an "alert" alias for long running commands. Use like so:
alias alert='notify-send --urgency=low -i "${? = 0 }" && echo terminal || echo
error)' "${history|tail -n1|sed -e '\s/[0-9]\+\s*//;s/;/&|]\s*alert$/'\
')"'
# ~/.bash_aliases, instead of adding them here directly.
if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
oscar@oscar-laptop:~$
```

怎么样，grep 命令很强大吧。如上图所见，grep 命令列出了 .bashrc 文件中所有包含 alias 的行，并且在终端中，以红色标出了每一个 alias。其实 grep 更像是一个过滤器，它可以筛选出我们要找的对象。

少年，莫激动，此对象非彼对象~

如果我们要用 grep 命令在一个文件中查找用空格隔开的文本，那么就要加上双引号，例如：

```
grep "Hello World" file2
```

-i 参数：忽略大小写

默认的情况下，grep 命令是区分大小写的，也就是说搜索的文本将严格按照大小写来搜索。比如我搜索的文本是 text，那么就不会搜出 Text，tExt，TEXT 等等文本。

但是我们可以给 grep 加上 -i 参数，使得 grep 可以忽略大小写。i 是英语 ignore 的缩写，表示“忽略”。

例如：

```
grep -i alias .bashrc
```

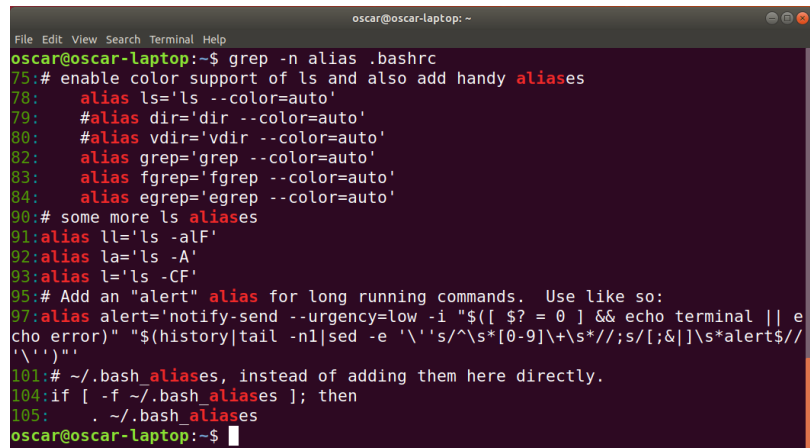
```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ grep -i alias .bashrc
# enable color support of ls and also add handy aliases
alias ls='ls --color=auto'
#alias dir='dir --color=auto'
#alias vdir='vdir --color=auto'
alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
# some more ls aliases
alias ll='ls -aF'
alias la='ls -A'
alias l='ls -CF'
# Add an "alert" alias for long running commands. Use like so:
alias alert='notify-send --urgency=low -i "${? = 0 }" && echo terminal || echo
error)' "${history|tail -n1|sed -e '\s/[0-9]\+\s*//;s/;/&|]\s*alert$/'\
')"'
# Alias definitions.
# ~/.bash_aliases, instead of adding them here directly.
if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
oscar@oscar-laptop:~$
```

可以看到，加了 -i 参数后，grep 的搜索结果就多了 **#Alias definitions**。那一行，因为 -i 参数使得 grep 搜索不区分大小写。

-n 参数：显示行号

-n 参数的作用很简单，就是显示搜索到的文本所在的行号。n 是英语 number 的缩写，表示“数字，编号”。

```
grep -n alias .bashrc
```

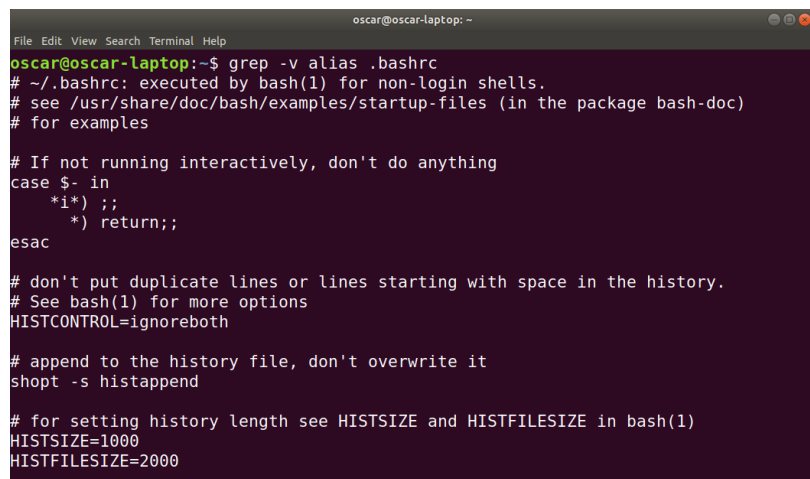


```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ grep -n alias .bashrc  
75: # enable color support of ls and also add handy aliases  
78: alias ls='ls --color=auto'  
79: #alias dir='dir --color=auto'  
80: #alias vdir='vdir --color=auto'  
82: alias grep='grep --color=auto'  
83: alias fgrep='fgrep --color=auto'  
84: alias egrep='egrep --color=auto'  
90: # some more ls aliases  
91: alias ll='ls -aF'  
92: alias la='ls -A'  
93: alias l='ls -CF'  
95: # Add an "alert" alias for long running commands. Use like so:  
97: alias alert='notify-send --urgency=low -i "${?} = 0" && echo terminal || echo error)' "${history|tail -n1|sed -e '\s/[0-9]\+\s*//;s/[;&]\s*alert$//\s')"  
101: # ~/.bash_aliases, instead of adding them here directly.  
104: if [ -f ~/.bash_aliases ]; then  
105: . ~/.bash_aliases  
oscar@oscar-laptop:~$
```

-v 参数：只显示文本不在的行

-v 参数很有意思，v 是 invert 的缩写，表示“颠倒，倒置”。-v 参数的作用与正常 grep 的作用正好颠倒，就是只显示搜索的文本不在的那些行。

```
grep -v alias .bashrc
```



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ grep -v alias .bashrc  
# ~/.bashrc: executed by bash(1) for non-login shells.  
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)  
# for examples  
  
# If not running interactively, don't do anything  
case $- in  
  *i*) ;;  
  *) return;;  
esac  
  
# don't put duplicate lines or lines starting with space in the history.  
# See bash(1) for more options  
HISTCONTROL=ignoreboth  
  
# append to the history file, don't overwrite it  
shopt -s histappend  
  
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)  
HISTSIZE=1000  
HISTFILESIZE=2000
```

可以看到，这次 grep 过滤出了 .bashrc 中所有不包含 alias 的行。

-r 参数：在所有子目录和子文件中查找

如果你不知道你要找的文本在哪个文件里，你可以用强大的 -r 参数。

r 是英语 recursive 的缩写，表示“递归”。

如果用了 -r 参数，那么 grep 命令使用时的最后一个参数（grep text file 这个模式中的 file）需要换成 directory，也就是必须是一个目录。因为 -r 参数是让 grep 命令能够在指定目录的所有子目录和子文件中查找文本。

例如：

```
grep -r "Hello World" folder/
```

表示在 `folder` 这个目录的所有子目录和子文件中查找 `Hello World` 这个文本。当然了，以上例子中，`folder` 后面的斜杠（/）不是必须的，这里只是为了清楚表明 `folder` 是一个目录。只要 `folder` 是一个目录，Linux 系统是不会搞错的。

Linux 中还有一个 `rgrep` 的命令，它的作用相当于 `grep -r`。

grep 的高级用法：配合正则表达式

正则表达式使用单个字符串来描述、匹配一系列符合某个句法规则的字符串。

`grep` 配合正则表达式就可以实现比较高级的搜索了。

我们首先来看一眼以下的这个表格，表格中列出了最常用的一些正则表达式的字符以及其含义：

特殊字符	含义
.	匹配除“\n”之外的任何单个字符
^	行首（匹配输入字符串的开始位置）
\$	行尾（匹配输入字符串的结束位置）
[]	在中括号中的任意一个字符
?	问号前面的元素出现零次或一次
*	星号前面的元素可能出现零次、一次或多次
+	加号前面的元素必须出现一次以上（包含一次）
	逻辑或
()	表达式的分组（表示范围和优先度）

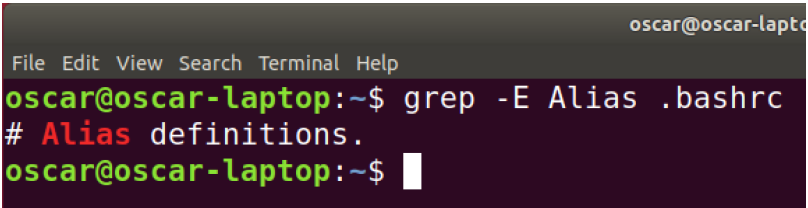
当然了，上表没有列出所有的正则表达式的字符。大家可以上网搜索，查找完整的表格。

看了上表你可能会说：“我啥也看不懂啊...”

这是正常的，正则表达式不是那么容易入门的，但也不是太难。要把正则表达式讲清楚，大概需要两课的篇幅，我们这里就不多做介绍了。

首先，为了让 `grep` 命令知道我们要使用正则表达式，须要加上 `-E` 参数（`E` 是 `extended regular expression` 的第一个字母，表示“扩展的正则表达式”）。例如：

```
grep -E Alias .bashrc
```



当然了，Linux 也有一个命令 `egrep`，其效果等同 `grep -E`。

不要怀疑，`Alias` 也算是一个正则表达式，只不过没有用到上面表格中的特殊符号而已。

到此为止，没什么新鲜的。我们用正则表达式只是和之前的搜索类似。接下来，我们才真的要用到正则表达式的特殊字符了。

首先来看这个例子：

```
grep -E ^alias .bashrc
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ grep -E ^alias .bashrc
alias ll='ls -aLF'
alias la='ls -A'
alias l='ls -CF'
alias alert='notify-send --urgency=low -i "${? = 0 }" && echo terminal || echo
error)' "${history|tail -n1|sed -e '\s/^[0-9]\+\s*//;s/[:&|]\s*alert$/'\s'
}'"
```

这个例子中，我们用到了 `^` 这个特殊符号，上面的表格里对于 `^` 已经做了说明：行首（匹配输入字符串的开始位置）。也就是说，`^` 后面的字符须要出现在一行的开始。

因此，就搜出了如上图中的四行，这四行都是包含 `alias`，并且以 `alias` 开头的。

再来举几个例子：

```
grep -E [Aa]lias .bashrc
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ grep -E [Aa]lias .bashrc
# enable color support of ls and also add handy aliases
alias ls='ls --color=auto'
#alias dir='dir --color=auto'
#alias vdir='vdir --color=auto'
alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
# some more ls aliases
alias ll='ls -aLF'
alias la='ls -A'
alias l='ls -CF'
# Add an "alert" alias for long running commands. Use like so:
alias alert='notify-send --urgency=low -i "${? = 0 }" && echo terminal || echo
error)' "${history|tail -n1|sed -e '\s/^[0-9]\+\s*//;s/[:&|]\s*alert$/'\s'
}'"
```

上面的表格里解释了 `[]` 的作用，是将 `[]` 中的字符任取其一，因此 `[Aa]lias` 的意思就是既可以是 `Alias`，又可以是 `alias`。因此 `grep` 的搜索结果把包含 `Alias` 和 `alias` 的行都列出来了。

再比如：

```
grep -E [0-4] .bashrc
```

用于搜索包含 0 至 4 的任一数字的行。

```
grep -E [a-zA-Z] .bashrc
```

用于搜索包含在 a 至 z 之间的任意字母或者 A 至 Z 之间的任意字母的行。

其他正则表达式还有很多例子。就不一一列举了。

注意：

其实在 Ubuntu 这样的 Linux 发行版中，`grep` 如果要和正则表达式配合，不加 `-E` 参数也是可以的，正则表达式始终是激活的。不过有的 Unix 发行版的系统可能不加 `-E` 参数就不能搜索正则表达式。因此为了兼容，我们一般来说还是加上 `-E` 参数比较好。

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ grep ^alias .bashrc
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias alert='notify-send --urgency=low -i "${? = 0} && echo terminal || echo error" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&]\s*alert$//'\`'
)'"
oscar@oscar-laptop:~$
```

上图中，我们可以看到，在 Ubuntu 这个 Linux 发行版中，grep 不加 -E 参数也能和正则表达式完美配合。

3. sort 命令：为文件排序

sort 是英语“排序”的意思。sort 命令用于对文件的行进行排序。

如果学过数据结构和算法，那应该对那几个 Sorting 算法有印象，快速排序（Quick Sort），归并排序（Merge Sort），插入排序（Insertion Sort），等等。

为了演示，我们首先用文本编辑器（可以用 nano）来创建一个文件，名叫 name.txt 比如，然后在里面写入以下的行：

```
John
Paul
Luc
Matthew
Mark
jude
Daniel
Samuel
Job
```

随便写几个英语常用名字就可以了。name 是英语“名字”的意思。

然后，我们用 sort 命令来举个例子：

```
sort name.txt
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop: ~
oscar@oscar-laptop:~$ sort name.txt
Daniel
Job
John
jude
Luc
Mark
Matthew
Paul
Samuel
oscar@oscar-laptop:~$
```

可以看到，sort 命令将 name.txt 文件中的行按照首字母的英文字典顺序进行了排列。

sort 命令并不区分大小写，小写字母开头的 jude 还是排在 John 之后。

-o 参数：将排序后的内容写入新文件

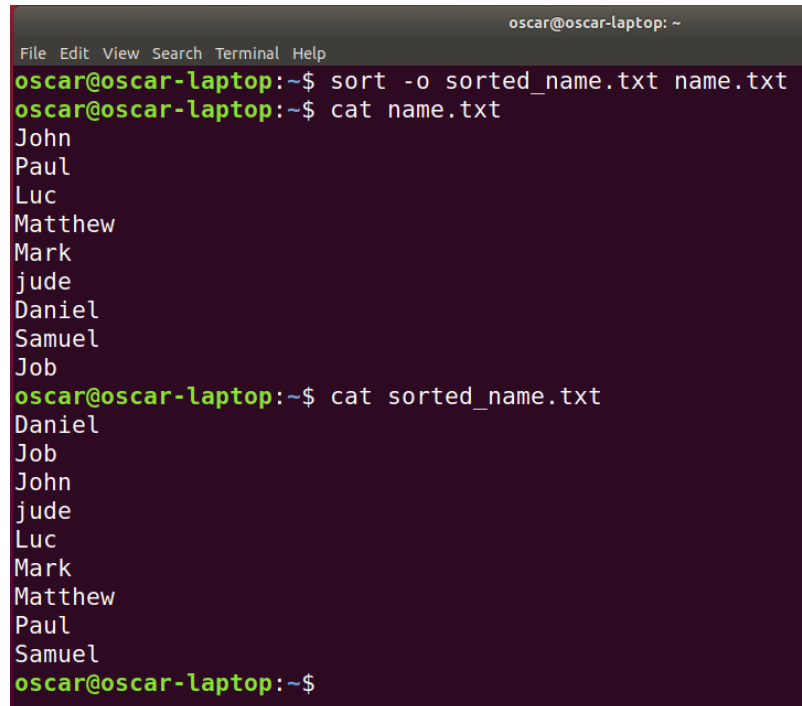
如果你打开 name.txt 文件，你会发现，经过了 sort 命令的“洗礼”，name.txt 中的内容还是维持原来的顺序。

单独使用 sort 命令是不会真正改变文件内容的，只是把排序结果显示在终端上。

那我们要存储排序结果到新的文件怎么办呢？可以用 -o 参数。

o 是 output 的首字母，表示“输出”，就是将排序结果输出到文件中。

```
sort -o name_sorted.txt name.txt
```



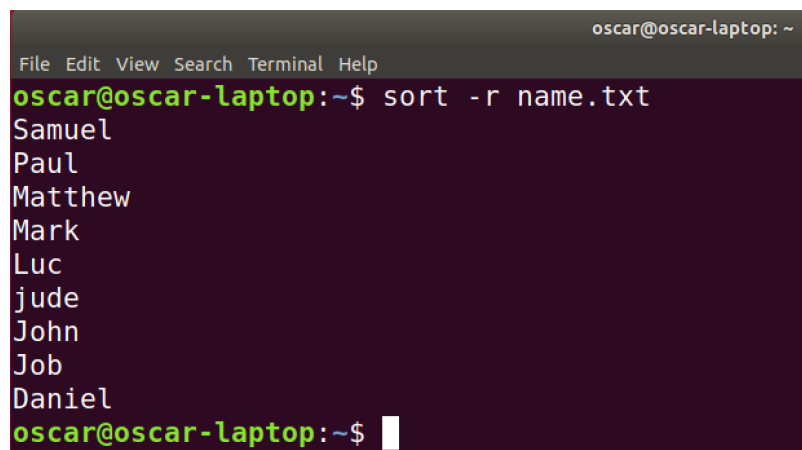
```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ sort -o sorted_name.txt name.txt  
oscar@oscar-laptop:~$ cat name.txt  
John  
Paul  
Luc  
Matthew  
Mark  
jude  
Daniel  
Samuel  
Job  
oscar@oscar-laptop:~$ cat sorted_name.txt  
Daniel  
Job  
John  
jude  
Luc  
Mark  
Matthew  
Paul  
Samuel  
oscar@oscar-laptop:~$
```

可以看到，name.txt 经过 sort 命令排序之后的内容被储存在了新的文件 name_sorted.txt 中，而 name.txt 的内容是不变的。

-r 参数：倒序排列

-r 参数中的 r 是 reverse 的缩写，是“相反”的意思，与普通的仅用 sort 命令正好相反。

```
sort -r name.txt
```



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ sort -r name.txt  
Samuel  
Paul  
Matthew  
Mark  
Luc  
jude  
John  
Job  
Daniel  
oscar@oscar-laptop:~$
```

-R 参数：随机排序

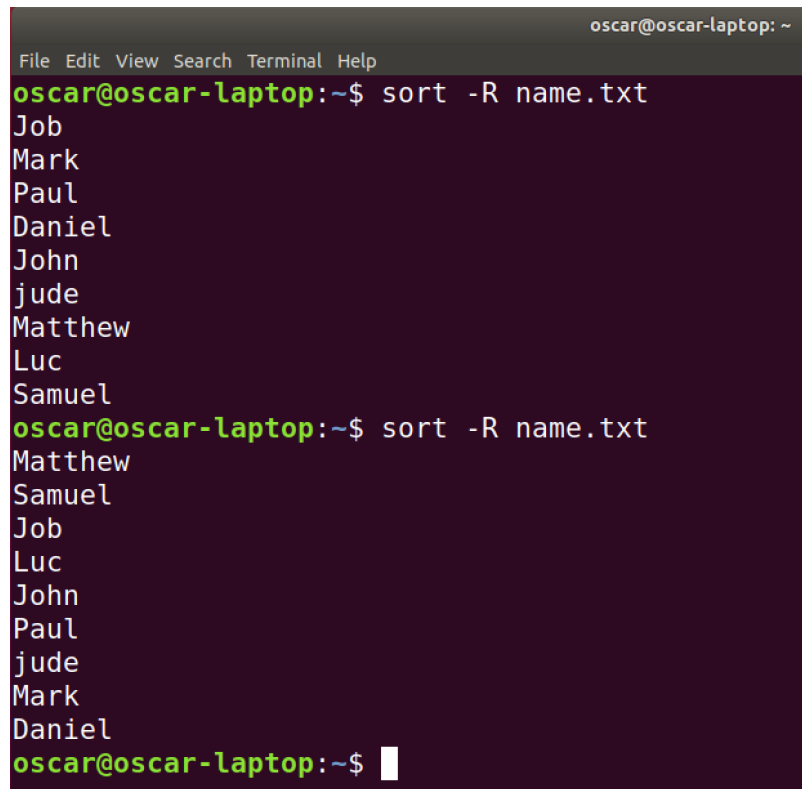
R 是英语 random 的首字母，表示“随机的，任意的”。

-R 参数比较“无厘头”，因为它会让 sort 命令的排序变为随机，就是任意排序，也许每次都不一样。

但在有些时候，-R 参数还是很有用的。

```
sort -R name.txt
```

为了显示每次排序都是随机的，我们将以上命令运行了两次：



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ sort -R name.txt  
Job  
Mark  
Paul  
Daniel  
John  
jude  
Matthew  
Luc  
Samuel  
oscar@oscar-laptop:~$ sort -R name.txt  
Matthew  
Samuel  
Job  
Luc  
John  
Paul  
jude  
Mark  
Daniel  
oscar@oscar-laptop:~$
```

-n 参数：对数字排序

对数字的排序有点特殊。默认仅用 sort 命令的时候，是不区分字符是否是数字的，会把这些数字看成字符串，按照 1-9 的顺序来排序。例如 138 会排在 25 前面，因为 1 排在 2 的前面。

那如果我们要 sort 命令识别整个数字，比如按照整个数值的大小顺序来说，25 应该排在 138 前面，那该怎么办呢？

就可以请出我们的 -n 参数了。n 是 number 的缩写。是英语“数字”的意思。-n 参数用于对数字进行排序，按从小到大排序。

为了演示，我们再用文本编辑器来创建一个文件，就叫 number.txt 好了。

里面随便填一些数字，每行一个：

```
12  
9  
216  
28  
174  
35  
68
```

然后用 `sort` 不加 `-n` 参数和加上 `-n` 参数分别测试：

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ sort number.txt
12
174
216
28
35
68
9
oscar@oscar-laptop:~$ sort -n number.txt
9
12
28
35
68
174
216
oscar@oscar-laptop:~$
```

可以看到，不加 `-n` 参数时，`sort` 就会把这些数字看成字符串，按字符依次来排序，按照 1-9 的顺序。

加上 `-n` 参数，就会把各行的数字看成一个整体，按照大小从小到大来排序了。

4. `wc` 命令：文件的统计

`wc` 是 word count 的缩写（此处 `wc` 不是“厕所”的意思...），`word` 是“单词”的意思，`count` 是“计算，统计”的意思。

因此，`wc` 命令看起来是用来统计单词数目的，但其实 `wc` 的功能不仅止于此。`wc` 命令还可以用来统计行数，字符数，字节数等。

跟前面的命令一样，`wc` 命令的用法也是后接文件名。`wc` 命令很有用，应该会成为你常用的命令之一。

如果不加选项参数，那么 `wc` 命令的返回值会有些特殊，有点晦涩难懂。

例如：

```
wc name.txt
```

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ wc name.txt
 9  9 50 name.txt
oscar@oscar-laptop:~$
```

可以看到返回值是

```
9 9 50 name.txt
```

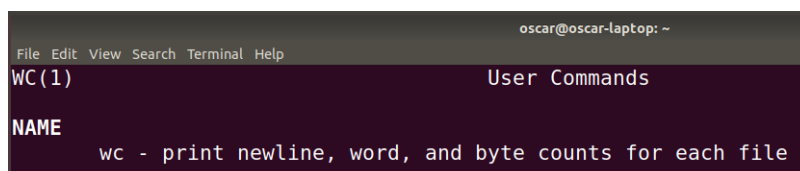
最后的 name.txt 只是表示文件名，不需考虑。

那么这三个数字：9，9，和 50 分别表示什么呢？

这三个数字，按顺序，分别表示：

- 行数（newline counts）：newline 是英语“换行、换行符”的意思。统计行数其实就是统计换行符的数目。
- 单词数（word counts）
- 字节数（byte counts）：byte 是英语“字节”的意思，等于 8 个二进制位（bit）。

可以用 man wc 查看 wc 的命令手册得知：



```
oscar@oscar-laptop: ~  
File Edit View Search Terminal Help  
wc(1) User Commands  
NAME  
wc - print newline, word, and byte counts for each file
```

wc 的命令描述是“print newline, word, and byte counts for each file”，翻成中文就是“对每个文件，打印其行数，单词数和字节数”。

因为我们之前创建 name.txt 时，每一行只有一个单词（英语名字），所以这里统计的行数和单词数都是 9。

50 代表字节数。我数了一下，name.txt 里的 9 个英语单词一共包含 41 个英语字母（也就是 41 个英语字符），占用 41 个字节。再加上每行结尾的换行符（Linux 中换行符是 '\n'），共有 9 个换行符，占用 9 个字节。41 + 9 = 50，正好是 50 个字节。

我们稍微讲一下字符和字节的一些联系和区别：

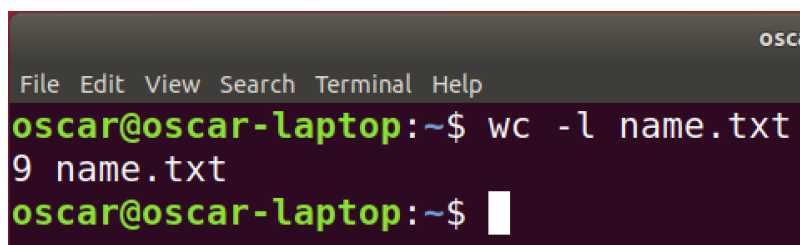
- 字节（Byte 或 Octet）是计量单位，表示数据量多少，是计算机存储容量的计量单位。一个字节等于 8 位（Bit，比特位，是计算机最小的存储单位。就是 0 或 1 这样的二进制位）。
- 字符（Character）是计算机中使用的文字和符号，比如“a”、“B”、“7”、“&”、“%”等。不同语言有不同的字符，一般我们中国人接触比较多的是英语和中文的字符。

字符在不同的编码中所占字节数是不一样的。字符的编码和标准有不少，这里我们就不深入展开了，大家可以看这个链接来深入了解：[字符集](#)。

-l 参数：统计行数

为了只统计行数，我们可以加上 -l 参数。l 是 line 的缩写，表示“行”。

```
wc -l name.txt
```

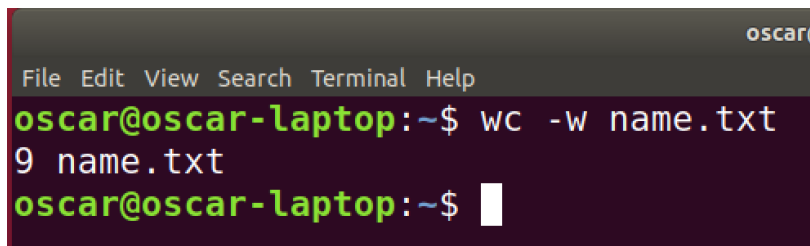


```
osca  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ wc -l name.txt  
9 name.txt  
oscar@oscar-laptop:~$
```

-w 参数：统计单词数

w 是 word 的缩写，表示“单词”。因此 -w 参数用于统计单词。

```
wc -w name.txt
```

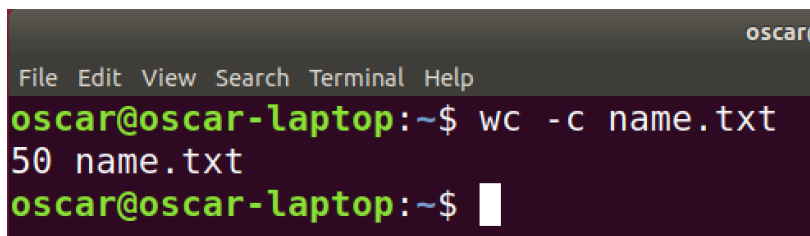


```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ wc -w name.txt
9 name.txt
oscar@oscar-laptop:~$
```

-c 参数：统计字节数

不知道什么是 c，因为 byte 或者 octet（都表示“字节”）的首字母都不是 c 啊。也许 c 是 character（英语“字符”的意思）的缩写吧。

```
wc -c name.txt
```

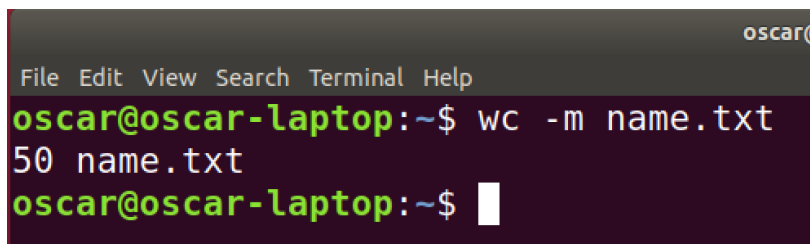


```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ wc -c name.txt
50 name.txt
oscar@oscar-laptop:~$
```

-m 参数：统计字符数

不知道什么是 m，因为 character（英语“字符”）的首字母不是 m：

```
wc -m name.txt
```



```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ wc -m name.txt
50 name.txt
oscar@oscar-laptop:~$
```

为了加深理解，我们来测试一下。创建一个只包含中文字符的文本文件，可以起名叫 chinese.txt（chinese 是“中文”的意思）。在里面写入：

```
你好吗
我很好
```

这 6 个汉字。

我们用 wc 命令来统计一下 chinese.txt 的字节数和字符数：

```
wc -c chinese.txt
```

```
wc -m chinese.txt
```

```

oscar@o
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ cat chinese.txt
你好吗
我很好
oscar@oscar-laptop:~$ wc -c chinese.txt
20 chinese.txt
oscar@oscar-laptop:~$ wc -m chinese.txt
8 chinese.txt
oscar@oscar-laptop:~$ █

```

可以看到，chinese.txt 包含的字节数是 20，字符数是 8。为什么呢？

其实这是因为使用的是 Unicode 标准 的 UTF-8 编码方式。中文字符占 3 个字节，一共有 6 个中文字符， $6 * 3 = 18$ ，再加上 2 个换行符占 2 个字节， $18 + 2 = 20$ 。

字符数则是 $6 + 2 = 8$ 个。

我们可以用 `file` 命令来确定文件的类型，运行：

file chinese.txt

```
File Edit View Search Terminal Help
oscar@oscar-laptop:~$ cat chinese.txt
你好吗
我很好
oscar@oscar-laptop:~$ wc -c chinese.txt
20 chinese.txt
oscar@oscar-laptop:~$ wc -m chinese.txt
8 chinese.txt
oscar@oscar-laptop:~$ file chinese.txt
chinese.txt: UTF-8 Unicode text
oscar@oscar-laptop:~$ file name.txt
name.txt: ASCII text
oscar@oscar-laptop:~$ cat name.txt
John
Paul
Luc
Matthew
Mark
jude
Daniel
Samuel
Job
oscar@oscar-laptop:~$
```

可以看到，chinese.txt 的编码是 UTF-8 Unicode，name.txt 的编码是 ASCII。

5. uniq 命令：删除文件中的重复内容

有时候，文件中包含重复的行，我们想要将重复的内容删除，

这时，uniq 命令就显得很有用了。

uniq 是英语 unique 的缩写，表示“独一无二的”。

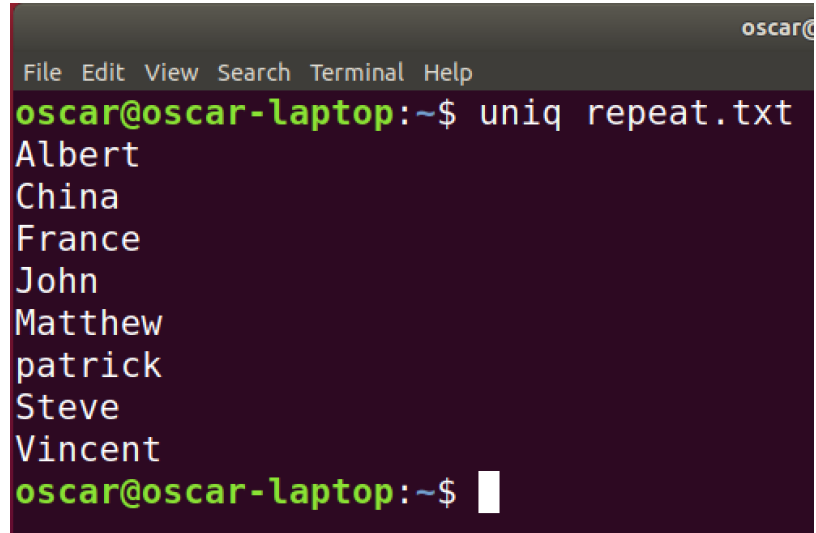
为了演示，我们创建一个文件 repeat.txt（repeat 是英语“重复”的意思），里面写入如下排序好的内容（因为 uniq 命令有点“呆”，只能将连续的重复行变为一行）：

```
Albert
China
France
France
France
John
Matthew
Matthew
patrick
Steve
Vincent
```

可以看到，有三个 France 连在一起，两个 Matthew 连在一起。

我们用 uniq 命令来处理看看：

```
uniq repeat.txt
```

A terminal window titled 'oscar@oscar-laptop' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'oscar@oscar-laptop:~\$'. The command 'uniq repeat.txt' has been executed, resulting in the following output: Albert, China, France, John, Matthew, patrick, Steve, Vincent. The prompt is now 'oscar@oscar-laptop:~\$' with a cursor.

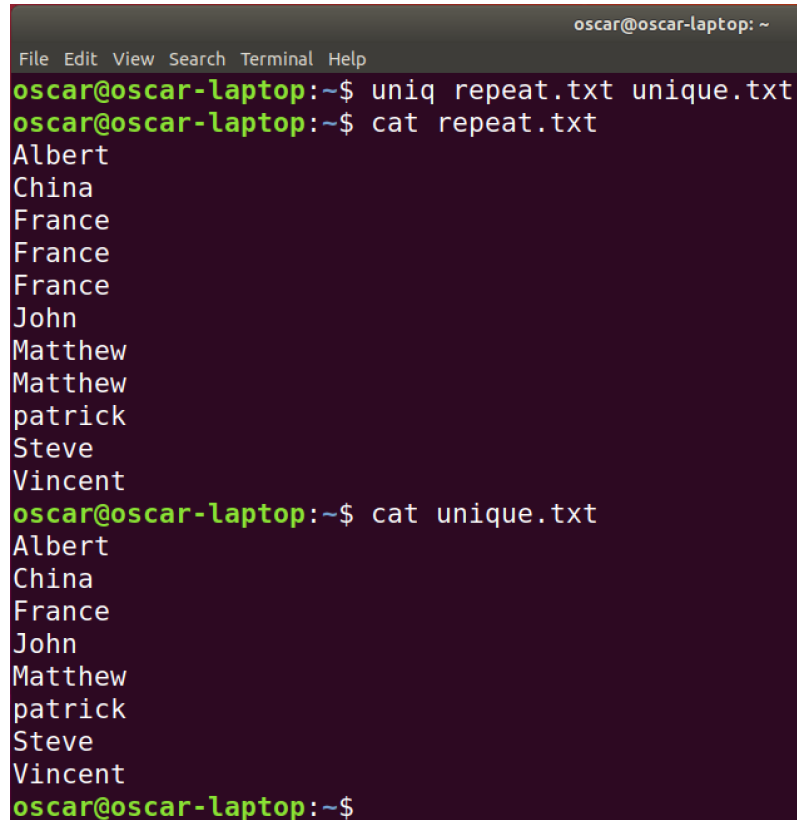
```
oscar@oscar-laptop:~$ uniq repeat.txt
Albert
China
France
John
Matthew
patrick
Steve
Vincent
oscar@oscar-laptop:~$
```

可以看到，三个连续的 France 只剩下一个了，两个连续的 Matthew 也只剩一个了。

和 sort 命令类似，uniq 命令并不会改变原文件的内容，只会把处理后的内容显示出来。

如果想将处理后的内容储存到一个新文件中，可以使用如下的方法：

```
uniq repeat.txt unique.txt
```

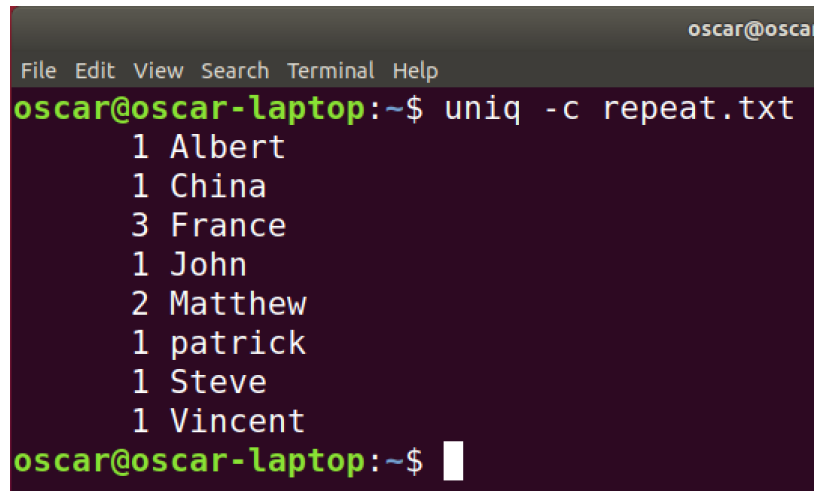
A terminal window titled 'oscar@oscar-laptop: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'oscar@oscar-laptop:~\$'. The command 'uniq repeat.txt unique.txt' has been executed. The prompt is now 'oscar@oscar-laptop:~\$'. The command 'cat repeat.txt' has been executed, resulting in the following output: Albert, China, France, France, France, John, Matthew, Matthew, patrick, Steve, Vincent. The prompt is now 'oscar@oscar-laptop:~\$'. The command 'cat unique.txt' has been executed, resulting in the following output: Albert, China, France, John, Matthew, patrick, Steve, Vincent. The prompt is now 'oscar@oscar-laptop:~\$'.

```
oscar@oscar-laptop:~$ uniq repeat.txt unique.txt
oscar@oscar-laptop:~$ cat repeat.txt
Albert
China
France
France
France
John
Matthew
Matthew
patrick
Steve
Vincent
oscar@oscar-laptop:~$ cat unique.txt
Albert
China
France
John
Matthew
patrick
Steve
Vincent
oscar@oscar-laptop:~$
```

-c 参数：统计重复的行数

-c 参数用于显示重复的行数，如果是独一无二的行，那么数目就是 1。c 是 count 的缩写，表示“统计，计数”。

```
uniq -c repeat.txt
```

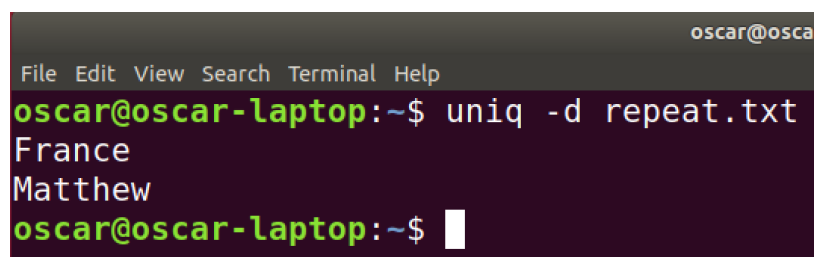


```
oscar@oscar-laptop:~$ uniq -c repeat.txt
 1 Albert
 1 China
 3 France
 1 John
 2 Matthew
 1 patrick
 1 Steve
 1 Vincent
oscar@oscar-laptop:~$
```

-d 参数：只显示重复行的值

-d 参数只显示重复的行的值。d 是 duplicated 的缩写，表示“重复的”。

```
uniq -d repeat.txt
```



```
oscar@oscar-laptop:~$ uniq -d repeat.txt
France
Matthew
oscar@oscar-laptop:~$
```

6. cut 命令：剪切文件的一部分内容

cut 是英语“剪切”的意思。大家平时肯定有剪切文本内容的经历吧，一般剪切之后还会把剪切的内容粘贴到某处。

cut 命令用于对文件的每一行进行剪切处理。

-c 参数：根据字符数来剪切

c 是 character 的缩写，表示“字符”。

比如，我们要 name.txt 的每一行只保留第 2 至第 4 个字符。可以这样做：

```
cut -c 2-4 name.txt
```



```
oscar@oscar-  
File Edit View Search Terminal Help  
oscar@oscar-laptop:~$ cut -c 2-4 name.txt  
ohn  
aul  
uc  
att  
ark  
ude  
ani  
amu  
ob  
oscar@oscar-laptop:~$
```

当然了，篇幅有限，我们不可能对每个命令的每个参数和每种用法都做详细介绍。大家可以用 `man` 命令来查询各个命令的手册，深入学习。

小结

1. `grep` 命令应该算是在文件中查找关键字最常用的工具了。
2. `grep` 命令可以通过正则表达式来查找。一开始正则表达式会比较难记，但是功能很强大。我们可以调用 `egrep` 命令，其等价于 `grep -E`。
3. `sort` 命令用于为文件中的行按字母顺序排序。使用 `-n` 参数可以按照数字顺序排序。
4. `wc` 命令可以统计文件中行数，单词数或者字节数。
5. `uniq` 命令可以用于删除文件中重复的内容。
6. `cut` 命令用于剪切文件的一部分内容。

今天的课就到这里，一起加油吧！