x6 x4

光环IT学院 AURA IT COLLEGE

-w：统计字数 这里应该是统计单词数
-help：显示帮助信息
--version：显示版本信息

一个汉字到底几个字节？

#1 p. 5

-r：倒序排序
-k：位置 1,位置 2 根据关键字排序，在从第位置 1 开始，位置 2 结束
-t：指定分隔符
-u：去重重复行
-o：将结果写入文件

不加-o的sort只是将排序后结果打印出来，而不会将原文件里的文本顺序改变，加-o会将排序后的结果写入原文件(但好像老是卡住，还不如将结果写入一个新文件,还能保留原文件，即：sort 文件名 > 新文件名。

准备数据：

```
aaa:10:1.1
ccc:20:3.3
bbb:40:4.4
```

#2 p. 6

```
/home/linux/txt/test.txt
/home/linux/txt/hw.txt
/home/linux/txt/sort.txt

## 忽略大小写查找文件名包含 linux        -i表示查找时忽略大小写
[linux@linux txt]$ find /home/linux/txt -iname "*linux*"
/home/linux/txt/LINUX.pdf

## 查找文件名结尾是.txt 或者.jpg 的文件   命令中用括号一定要用反斜杠转义，并且括号里的内容和左右括号之间一定要空一个空格
[linux@linux txt]$ find /home/linux/txt/ \( -name "*.txt" -o -name "*.jpg" \)
/home/linux/txt/liujialing.jpg
/home/linux/txt/uniq.txt
/home/linux/txt/mingxing.txt
```

#3 p. 20

Shell 操作实用技巧

目录

1、Shell 操作日期时间

date - print or set the system date and time
linux 系统为我们提供了一个命令 date，专门用来显示或者设置系统日期时间的。
语法格式为：
date [OPTION]... [+FORMAT]　　或者
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
常用的可选项有：
--help：显示辅助信息
--version：显示 date 命令版本信息
-u：显示目前的格林威治时间
-d：做日期时间相关的运算
--date='-dateStr'：做日期时间的相关运算

1、 显示系统当前日期时间

```
[root@hadoop ~]# date
```

以指定格式显示日期时间

```
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S'
```

2、 设置系统日期时间

```
[root@hadoop ~]# date -s "2017-01-01 01:01"
```

3、 有时候，我们操作日期时间，经常会要获取前几天或者后几天的时间，那么 date 命令也给我们提供了实现这个功能的可选项'-d'和'--date'，请看下面细细的例子

先看'-d'：

```
## 获取下一天的时间
[root@hadoop ~]# date -d next-day '+%Y-%m-%d %H:%M:%S'
[root@hadoop ~]# date -d 'next day' '+%Y-%m-%d %H:%M:%S'
另外一种写法：
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S' -d tomorrow

## 获取上一天的时间
[root@hadoop ~]# date -d last-day '+%Y-%m-%d %H:%M:%S'
另外一种写法：
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S' -d yesterday

## 获取下一月的时间
[root@hadoop ~]# date -d next-month '+%Y-%m-%d %H:%M:%S'

## 获取上一月的时间
[root@hadoop ~]# date -d last-month '+%Y-%m-%d %H:%M:%S'

## 获取下一年的时间
[root@hadoop ~]# date -d next-year '+%Y-%m-%d %H:%M:%S'

## 获取上一年的时间
[root@hadoop ~]# date -d last-year '+%Y-%m-%d %H:%M:%S'

## 获取上一周的日期时间：
[root@hadoop ~]# date -d next-week '+%Y-%m-%d %H:%M:%S'
[root@hadoop ~]# date -d next-monday '+%Y-%m-%d %H:%M:%S'
[root@hadoop ~]# date -d next-thursday '+%Y-%m-%d %H:%M:%S'
```

那么类似的，其实，last-year，last-month，last-day，last-week，last-hour，last-minute，last-second 都有对应的实现。相反的，last 对应 next，自己可以根据实际情况灵活组织

接下来，我们**来看'--date'，**它帮我实现任意时间前后的计算，来看具体的例子：

```
## 获取一天以后的日期时间
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S' --date='1 day'
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S' --date='-1 day ago'

## 获取一天以前的日期时间
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S' --date='-1 day'
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S' --date='1 day ago'
```

上面的例子显示出来了使用的格式，使用精髓在于改变前面的字符串显示格式，改变数

据，改变要操作的日期对应字段，除了天也有对应的其他实现：year，month，week，day，hour，minute，second，monday（星期，七天都可）

4、 date 能用来显示或设定系统的日期和时间，在显示方面，使用者能设定欲显示的格式，格式设定为一个加号后接数个标记，其中可用的标记列表如下：
使用范例：

```
[root@hadoop ~]# date '+%Y-%m-%d %H:%M:%S'
```

日期方面：

```
%a : 星期几 (Sun..Sat)
%A : 星期几 (Sunday..Saturday)
%b : 月份 (Jan..Dec)
%B : 月份 (January..December)
%c : 直接显示日期和时间
%d : 日 (01..31)
%D : 直接显示日期 (mm/dd/yy)
%h : 同 %b
%j : 一年中的第几天 (001..366)
%m : 月份 (01..12)
%U : 一年中的第几周 (00..53) (以 Sunday 为一周的第一天的情形)
%w : 一周中的第几天 (0..6)
%W : 一年中的第几周 (00..53) (以 Monday 为一周的第一天的情形)
%x : 直接显示日期 (mm/dd/yyyy)
%y : 年份的最后两位数字 (00.99)
%Y : 完整年份 (0000..9999)
```

时间方面：

```
%%: 打印出%
%n : 下一行
%t : 跳格
%H : 小时(00..23)
%k : 小时(0..23)
%l : 小时(1..12)
%M : 分钟(00..59)
%p : 显示本地 AM 或 PM
%P : 显示本地 am 或 pm
%r : 直接显示时间(12 小时制，格式为 hh:mm:ss [AP]M)
%s : 从 1970 年 1 月 1 日 00:00:00 UTC 到目前为止的秒数
%S : 秒(00..61)
%T : 直接显示时间(24 小时制)
%X : 相当于%H:%M:%S %p
%Z : 显示时区
```

若是不以加号作为开头，则表示要设定时间，而时间格式为 MMDDhhmm[[CC]YY][.ss]

```
MM 为月份，
DD 为日，
```

hh 为小时，
mm 为分钟，
CC 为年份前两位数字，
YY 为年份后两位数字，
ss 为秒数

例子：date "050602032017.55"

```
[root@hadoop04 hadoop]# date "050602032017.55"
Sat May  6 02:03:55 CST 2017
[root@hadoop04 hadoop]# date "+%Y-%m-%d %H:%M:%S"
2017-05-06 02:03:55
[root@hadoop04 hadoop]#
```

5、 有用的小技巧

```
## 获取相对某个日期前后的日期：
[root@hadoop ~]# date -d 'may 14 -2 weeks'

## 把时间当中无用的 0 去掉，比如：01:02:25 会变成 1:2:25
[root@hadoop ~]# date '+%-H:%-M:%-S'

## 显示文件最后被更改的时间
[root@hadoop ~]# date "+%Y-%m-%d %H:%M:%S" -r bin/removeJDK.sh

## 求两个字符串日期之间相隔的天数
[root@hadoop ~]#
expr '(' $(date +%s -d "2016-08-08") - $(date +%s -d "2016-09-09") ')' / 86400
expr `expr $(date +%s -d "2016-08-08") - $(date +%s -d "2016-09-09")` / 86400

## shell 中加减指定间隔单位
[root@hadoop ~]# A=`date +%Y-%m-%d`
[root@hadoop ~]# B=`date +%Y-%m-%d -d "$A +48 hours"`
```

2、高级文本处理命令

5.1、wc

功能： 统计文件行数、字节、字符数
常用选项：
-c：统计文件字节数，一个英文字母 1 字节，一个汉字占 2-4 字节（根据编码）
-m：统计文件字符数，一个英文字母 1 字符，一个汉字占 1 个字符
-l：统计多少行
-L：统计最长行的长度， 也可以统计字符串长度

-w：统计字数 这里应该是统计单词数
-help：显示帮助信息
--version：显示版本信息

一个汉字到底几个字节？

占 2 个字节的：〇
占 3 个字节的：基本等同于 GBK，含 21000 多个汉字
占 4 个字节的：中日韩超大字符集里面的汉字，有 5 万多个
一个 utf8 数字占 1 个字节
一个 utf8 英文字母占 1 个字节

示例：

统计文件信息

[linux@linux ~]$ **wc wc.txt**

4 8 77 mingxing.txt

行数 单词数 字节数 文件名

```
[root@hadoop04 shell]# ll
total 8
-rw-r--r--. 1 root root 86 Jun 15 22:58 wc1.txt
-rw-r--r--. 1 root root 77 Jun 15 22:51 wc.txt
[root@hadoop04 shell]# wc *
  4   9  86 wc1.txt
  4   8  77 wc.txt
  8  17 163 total
[root@hadoop04 shell]#
```

统计字符串长度

[linux@linux ~]$ **echo "hello" | wc -L**

5

```
[root@hadoop04 shell]# wc * -L
 37 wc1.txt
 28 wc.txt
 37 total
[root@hadoop04 shell]# ll
total 8
-rw-r--r--. 1 root root 86 Jun 15 22:58 wc1.txt
-rw-r--r--. 1 root root 77 Jun 15 22:51 wc.txt
[root@hadoop04 shell]#
```

统计文件行数：

[linux@linux ~]$ **wc -l mingxing.txt**

6 mingxing.txt

```
[root@hadoop04 shell]# ll
total 8
-rw-r--r--. 1 root root 86 Jun 15 22:58 wc1.txt
-rw-r--r--. 1 root root 77 Jun 15 22:51 wc.txt
[root@hadoop04 shell]# wc * -l
  4 wc1.txt
  4 wc.txt
  8 total
[root@hadoop04 shell]#
```

统计文件字数：

```
[linux@linux ~]$ wc -w mingxing.txt
7 mingxing.txt
```

5.2、sort

功能：排序文本，默认对整列有效

常用可选项：
-f：忽略字母大小写，就是将小写字母视为大写字母排序
-M：根据月份比较，比如 JAN、DEC
-h：根据易读的单位大小比较，比如 2K、1G
-g：按照常规数值排序
-n：根据字符串数值比较
-r：倒序排序
-k：位置 1,位置 2 根据关键字排序，在从第位置 1 开始，位置 2 结束
-t：指定分隔符
-u：去重重复行
-o：将结果写入文件

不加-o的sort只是将排序后结果打印出来，而不会将原文件里的文本顺序改变，加-o会将排序后的结果写入原文件(但好像老是卡住，还不如将结果写入一个新文件,还能保留原文件，即：sort 文件名 > 新文件名。

准备数据：

```
aaa:10:1.1
ccc:20:3.3
bbb:40:4.4
eee:40:5.5
ddd:30:3.3
bbb:40:4.4
fff:30:2.2
```

示例：

```
[linux@linux ~]$ cat sort.txt              ## 准备排序文件，查看该内容
aaa:10:1.1
ccc:20:3.3
bbb:40:4.4
eee:40:5.5
ddd:30:3.3
bbb:40:4.4
fff:30:2.2

[linux@linux ~]$ sort sort.txt             ## 直接排序，把整行当做一列字符串，字典顺序
aaa:10:1.1
bbb:40:4.4
bbb:40:4.4
ccc:20:3.3
```

```
ddd:30:3.3
eee:40:5.5
fff:30:2.2

[linux@linux ~]$ sort -nk 2 -t : sort.txt    ##  以:作为分隔符，取第二个字段按照数值进行排序
aaa:10:1.1
ccc:20:3.3
fff:30:2.2
ddd:30:3.3
bbb:40:4.4
bbb:40:4.4
eee:40:5.5

[linux@linux ~]$ sort -nk 2 -u -t : sort.txt       ##  和上一个不一样的是-u 为了去重
aaa:10:1.1
ccc:20:3.3
ddd:30:3.3
bbb:40:4.4

多列排序：以:分隔，按第二列数值排倒序，第三列正序
[linux@linux ~]$ sort -n -t: -k2,2r -k3 sort.txt
bbb:40:4.4
bbb:40:4.4
eee:40:5.5
fff:30:2.2
ddd:30:3.3
ccc:20:3.3
aaa:10:1.1
```

5.3、uniq

功能：去除重复行，只会统计相邻的
常用选项：
-c：打印出现的次数
-d：只打印重复行
-u：只打印不重复行
-D：只打印重复行，并且把所有重复行打印出来
-f N：比较时跳过前 N 列
-i：忽略大小写
-s N：比较时跳过前 N 个字符
-w N：对每行第 N 个字符以后内容不做比较

准备数据：

```
abc
xyz
cde
cde
xyz
abd
```

示例 1：

```
[linux@linux ~]$ uniq uniq.txt                  ## 直接去重，只能在相邻行去重
abc
xyz
cde
xyz
abd

[linux@linux ~]$ sort uniq.txt | uniq           ## 先给文件排序，然后去重
abc
abd
cde
xyz

[linux@linux ~]$ sort uniq.txt | uniq -c        ## 打印每行重复次数
      1 abc
      1 abd
      2 cde
      2 xyz

[linux@linux ~]$ sort uniq.txt | uniq -u -c     ## 打印不重复行，并给出次数
      1 abc
      1 abd

[linux@linux ~]$ sort uniq.txt | uniq -d -c     ## 打印重复行，并给出次数
      2 cde
      2 xyz

[linux@linux ~]$ sort uniq.txt | uniq -w 2      ## 以开头前两个字符为判断标准去重
abc
cde
xyz
```

示例 2：

先准备两个文件：a.txt　和　b.txt
文件内容分别为：

```
[hadoop@hadoop04 data]$ cat a.txt
a
b
c
d
[hadoop@hadoop04 data]$ cat b.txt
c
d
e
f
[hadoop@hadoop04 data]$
```

需求：

1、 求两个文件的交集：

[hadoop@hadoop04 data]$ **cat a.txt b.txt | sort | uniq -d**

```
[hadoop@hadoop04 data]$ cat a.txt b.txt | sort | uniq -d
c
d
```

2、 求两个文件的并集：

[hadoop@hadoop04 data]$ **cat a.txt b.txt | sort | uniq**

```
[hadoop@hadoop04 data]$ cat a.txt b.txt | sort | uniq
a
b
c
d
e
f
```

3、 求 a.txt 和 b.txt 的差集

[hadoop@hadoop04 data]$ **cat a.txt b.txt b.txt | sort | uniq -u**

```
[hadoop@hadoop04 data]$ cat a.txt b.txt b.txt | sort | uniq -u
a
b
```

4、 求 b.txt 和 a.txt 的差集

[hadoop@hadoop04 data]$ **cat b.txt a.txt a.txt | sort | uniq -u**

```
[hadoop@hadoop04 data]$ cat b.txt a.txt a.txt | sort | uniq -u
e
f
```

2.4、cut

cut 命令可以从一个文本文件或者文本流中提取文本列

cut 语法

cut -d'分隔字符' -f fields　　## 用于有特定分隔字符

cut -c 字符区间　　## 用于排列整齐的信息

选项与参数：

-d：后面接分隔字符。与 -f 一起使用

-f：依据 -d 的分隔字符将一段信息分割成为数段，用 -f 取出第几段的意思
-c：按照字符截取
-b：按照字节截取

例子 1：

首先看 PATH 变量：
[root@localhost ~]# **echo $PATH**
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin

将 PATH 变量取出，找出第五个路径
[root@localhost ~]# **echo $PATH | cut -d ':' -f 5**
/usr/sbin

将 PATH 变量取出，找出第三和第五个路径，以下三种方式都 OK
[root@localhost ~]# **echo $PATH | cut -d ':' -f 3,5**
[root@localhost ~]# **echo $PATH | cut -d : -f 3,5**
[root@localhost ~]# **echo $PATH | cut -d: -f3,5**
/sbin:/usr/sbin

将 PATH 变量取出，找出第三到最后一个路径
[root@localhost ~]# **echo $PATH | cut -d ':' -f 3-**
/sbin:/bin:/usr/sbin:/usr/bin:/root/bin

将 PATH 变量取出，找出第一到第三，还有第五个路径
[root@localhost ~]# **echo $PATH | cut -d ':' -f 1-3,5**
/usr/local/sbin:/usr/local/bin:/sbin:/usr/sbin

例子 2：

先准备已空格分开的这么段数据：
黄渤 huangbo 18 jiangxi
徐峥 xuzheng 22 hunan
王宝强 wangbaoqiang 44 liujiayao

获取中间的年龄：
[root@localhost ~]# **cut -f 3 -d ' ' cut.txt**
18
22
44

获取第二个字符到第五个字符之间的字符：
[root@localhost ~]# **cut -c 2-5 cut.txt**
渤 hu
峥 xu
宝强 w

获取第四个字节到第六个字节中的字符：
[root@hadoop ~]# **cut -b 4-6 cut.txt**
渤
峥
宝

5.5、grep（文本生成器）

grep 是一种强大的文本搜索工具，他能使用正则表达式搜索文本，并把匹配的行统计出来

命令：grep [选项] [– color=auto] ”搜索字符串” filename
常用参数：
-c：统计符合条件的字符串出现的总行数。
-E：支持扩展正则表达式。
-i：忽略字符大小写。
-n：在显示匹配到的字符串前面加上行号。
-v：显示没有”搜索字符串”内容的那一行。
-l：列出文件内容中有搜索字符串的文件名称。
-o：只输出文件中匹配到的部分。
-color=auto：将匹配到的字符串高亮出来。

1、基本使用
查询包含 hadoop 的行
grep hadoop /etc/passwd

```
[root@localhost ~]# grep hadoop /etc/passwd
hadoop:x:500:504:hadoop01:/home/hadoop:/bin/bash
```

grep huangbo ./*.txt ## 寻找当前路径下所有 txt 当中内容那些是带了 huangbo 字符串的

```
[root@localhost ~]# grep huangbo ./*.txt
./mazhonghua.txt:my name is huangbo is is huangbo
./sutdent.txt:huangbo 18 jiangxi
```

2、先看一份数据：grep.txt

```
huangbo is shuaige
huangxiaoming is shuaige
liuyifei is meinv
hello world hello tom hello kitty

#how old are you
#one two three four five six seven eight nine ten
```

2.1、统计出现某个字符串的行的总行数

grep -c 'hello' grep.txt

grep -c 'is' grep.txt

[hadoop@hadoop04 ~]# **grep -c 'hello' grep.txt**

[hadoop@hadoop04 ~]# **grep -c 'is' grep.txt**

```
[hadoop@hadoop04 myshell]$ grep -c 'hello' grep.txt
1
[hadoop@hadoop04 myshell]$ grep -c 'is' grep.txt
3
[hadoop@hadoop04 myshell]$
```

2.2、查询不包含 is 的行

grep -v 'is' grep.txt

[hadoop@hadoop04 ~]# **grep -v 'is' grep.txt**

```
[hadoop@hadoop04 myshell]$  grep -v 'is' grep.txt
hello world hello tom hello kitty

#how old are you
#one two three four five six seven eight nine ten
```

2.3、正则表达包含 huang

grep '.*huang.*' grep.txt

[hadoop@hadoop04 ~]# **grep '.*huang.*' grep.txt**

```
[hadoop@hadoop04 myshell]$ grep '.*huang.*' grep.txt
huangbo is shuaige
huangxiaoming is shuaige
[hadoop@hadoop04 myshell]$
```

2.4、输出匹配行的前后 N 行（会包括匹配行）

使用-A 参数输出匹配行的后一行：grep -A 1 "huangxiaoming" grep.txt

使用-B 参数输出匹配行的前一行：grep -B 1 "huangxiaoming" grep.txt

使用-C 参数输出匹配行的前后各一行：grep -C 1 "huangxiaoming" grep.txt

```
[hadoop@hadoop04 myshell]$ cat grep.txt
huangbo is shuaige
huangxiaoming is shuaige
liuyifei is meinv
hello world hello tom hello kitty

#how old are you
#one two three four five six seven eight nine ten
[hadoop@hadoop04 myshell]$ grep -A 1 "huangxiaoming" grep.txt
huangxiaoming is shuaige
liuyifei is meinv
[hadoop@hadoop04 myshell]$ grep -B 1 "huangxiaoming" grep.txt
huangbo is shuaige
huangxiaoming is shuaige
[hadoop@hadoop04 myshell]$ grep -C 1 "huangxiaoming" grep.txt
huangbo is shuaige
huangxiaoming is shuaige
liuyifei is meinv
[hadoop@hadoop04 myshell]$
```

3、正则表达(点代表任意一个字符)

grep 'h.*p' /etc/passwd

4、正则表达以 hadoop 开头

grep '^hadoop' /etc/passwd

5、正则表达以 hadoop 结尾

grep 'hadoop$' /etc/passwd

以 h 或 r 开头的

grep '^[hr]' /etc/passwd

不是以 h 和 r 开头的

grep '^[^hr]' /etc/passwd

不是以 h 到 r 开头的

grep '^[^h-r]' /etc/passwd

正则表达式的简单规则：

. ：任意一个字符

a*：任意多个 a(零个或多个 a)

a?：零个或一个 a

a+：一个或多个 a

.*：任意多个任意字符

\.：转义.

o\{2\}：o 重复两次

[A-Z]

[ABC]

查找不是以#开头的行

grep -v '^#' grep.txt | grep -v '^$'

[root@localhost ~]# **grep -v '^#' grep.txt**

```
[hadoop@hadoop04 myshell]$ grep -v '^#' grep.txt
huangbo is shuaige
huangxiaoming is shuaige
liuyifei is meinv
hello world hello tom hello kitty

[hadoop@hadoop04 myshell]$
```

[root@localhost ~]# **grep -v '^#' grep.txt | grep -v '^$'**

```
[hadoop@hadoop04 myshell]$ grep -v '^#' grep.txt | grep -v '^$'
huangbo is shuaige
huangxiaoming is shuaige
liuyifei is meinv
hello world hello tom hello kitty
```

5.6、sed（流编辑器）

sed 叫做流编辑器，在 shell 脚本和 Makefile 中作为过滤一使用非常普遍，也就是把前一个程序的输出引入 sed 的输入，经过一系列编辑命令转换成为另一种格式输出。sed 是一种在线编辑器，它一次处理一行内容，处理时，把当前处理的行存储在临时缓冲区中，称为"模式空间",接着用 sed 命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出。

选项:
-n：一般 sed 命令会把所有数据都输出到屏幕，如果加入-n 选项的话，则只会把经过 sed 命令处理的行输出到屏幕。
-e：允许对输入数据应用多条 sed 命令编辑。
-i：用 sed 的修改结果直接修改读取数据的文件，而不是由屏幕输出。

动作:
a：追加，在当前行后添加一行或多行。
c：行替换，用 c 后面的字符串替换原数据行。
i：插入，在当前行前插入一行或多行。
p：打印，输出指定的行。
s：字符串替换，用一个字符串替换另外一个字符串。格式为'行范围 s/旧字符串/新字符串/g'(如果不加 g 的话，则表示只替换每行第一个匹配的串)

1、删除：d 命令

sed '2d' sed.txt -----删除 sed.**txt** 文件的第二行。
sed '2,$d' sed.txt -----删除 sed.**txt** 文件的第二行到末尾所有行。
sed '$d' sed.txt -----删除 sed.**txt** 文件的最后一行。
sed '/test/d ' sed.txt -----删除 sed.**txt** 文件所有包含 test 的行。
sed '/[A-Za-z]/d ' sed.txt -----删除 sed.**txt** 文件所有包含字母的行。

2、整行替换：c 命令

将第二行替换成 hello world
sed '2c hello world' sed.txt

3、字符串替换：s 命令

sed 's/hello/hi/g' sed.txt
在整行范围内把 hello 替换为 hi。如果没有 g 标记，则只有每行第一个匹配的 hello 被替换成 hi。

sed 's/hello/hi/2' sed.txt
此种写法表示只替换每行的第 2 个 hello 为 hi

sed 's/hello/hi/2g' sed.txt

此种写法表示只替换每行的第 2 个以后的 hello 为 hi（包括第 2 个）

sed -n 's/^hello/hi/p' sed.txt

(-n)选项和 p 标志一起使用表示只打印那些发生替换的行。也就是说，如果某一行开头的 hello 被替换成 hi，就打印它。

sed -n '2,4p' sed.txt

打印输出 sed.txt 中的第 2 行和第 4 行

sed -n 's/hello/&-hi/gp' sed.txt

sed 's/^192.168.0.1/&-localhost/' sed.txt

sed 's/^192.168.0.1/[&]/' sed.txt

&符号表示追加一个串到找到的串后。所有以 192.168.0.1 开头的行都会被替换成它自己加 -localhost，变成 192.168.0.1-localhost。第三句表示给 IP 地址添加中括号

sed -n 's/\(liu\)jialing/\1tao/p' sed.txt

sed -n 's/\(liu\)jia\(ling\)/\1tao\2ss/p' sed.txt

liu 被标记为\1，所以 liu 会被保留下来（\1 == liu）

ling 被标记为\2，所以 ling 也会被保留下来（\2 == ling）

所以最后的结果就是\1tao\2ss == "liu" + "tao" + "ling" + "ss"

此处切记：\1 代表的是被第一个()包含的内容，\1 代表的是被第一个()包含的内容，……

上面命令的意思就是：被括号包含的字符串会保留下来，然后跟其他的字符串比如 tao 和 ss 组成新的字符串 liutaolingss

sed 's#hello#hi#g' sed.txt

不论什么字符，紧跟着 s 命令的都被认为是新的分隔符，所以，"#"在这里是分隔符，代替了默认的"/"分隔符。表示把所有 hello 替换成 hi。

选定行的范围：逗号

sed -n '/today/,/hello/p' sed.txt

所有在模板 today 和 hello 所确定的范围内的行都被打印。都找第一个，也就是说，从第一个 today 到第一个 hello

sed -n '5,/^hello/p' sed.txt

sed -n '/^hello/,8p' sed.txt

打印从第五行开始到第一个包含以 hello 开始的行之间的所有行。

sed '/today/,/hello/s/$/www/' sed.txt

对于模板 today 和 hello 之间的行，每行的末尾用字符串 www 替换。

sed '/today/,/hello/s/^/www/' sed.txt

对于模板 today 和 hello 之间的行，每行的开头用字符串 www 替换。

sed '/^[A-Za-z]/s/5/five/g' sed.txt

将以字母开头的行中的数字 5 替换成 five

4、多点编辑：e 命令

sed -e '1,5d' -e 's/hello/hi/' sed.txt

(-e)选项允许在同一行里执行多条命令。如例子所示，第一条命令删除 1 至 5 行，第二条命令用 hello 替换 hi。命令的执行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。

sed --expression='s/hello/hi/' --expression='/today/d' sed.txt

一个比-e 更好的命令是--expression。它能给 sed 表达式赋值。

5、从文件读入：r 命令

sed '/hello/r file' sed.txt

file 里的内容被读进来，显示在与 hello 匹配的行下面，如果匹配多行，则 file 的内容将显示在所有匹配行的下面。

6、写入文件：w 命令

sed -n '/hello/w file' sed.txt

在 huangbo.txt 中所有包含 hello 的行都被写入 file 里。

7、追加命令：a 命令

sed '/^hello/a\\--->this is a example' sed.txt

'--->this is a example'被追加到以 hello 开头的行(另起一行)后面，sed 要求命令 a 后面有一个反斜杠。

8、插入：i 命令

sed '/will/i\\some thing new ----------' sed.txt

如果 test 被匹配，则把反斜杠后面的文本插入到匹配行的前面。

9、下一个：n 命令

sed '/hello/{n; s/aa/bb/;}' sed.txt **替换下一行的第一个 aa**

sed '/hello/{n; s/aa/bb/g;}' sed.txt **替换下一行的全部 aa**

如果 hello 被匹配，则移动到匹配行的下一行，替换这一行的 aa，变为 bb，并打印该行，然后继续。

10、退出：q 命令

sed '10q' sed.txt

打印完第 10 行后，退出 sed。

同样的写法：

sed -n '1,10p ' sed.txt

5.7、awk（报表生成器）

Awk 是一个强大的处理文本的编程语言工具，其名称得自于它的创始人 Alfred Aho、Peter

Weinberger 和 Brian Kernighan 姓氏的首个字母，相对于 grep 的查找，sed 的编辑，awk 在其对数据分析并生成报告时，显得尤为强大。AWK 提供了极其强大的功能：可以进行样式装入、流控制、数学运算符、进程控制语句甚至于内置的变量和函数。简单来说 awk 就是扫描文件中的每一行，查找与命令行中所给定内容相匹配的模式。如果发现匹配内容，则进行下一个编程步骤。如果找不到匹配内容，则继续处理下一行。

1、假设 last -n 5 的输出如下:

```
[root@localhost ~]# last -n 5
root     pts/0        192.168.123.1    Wed Dec 28 01:55   still logged in
reboot   system boot  2.6.32-573.el6.x Tue Dec 27 04:25 - 03:11  (22:46)
root     pts/1        192.168.123.1    Tue Dec 27 02:00 - 02:00  (00:00)
root     pts/1        192.168.123.1    Tue Dec 27 01:59 - 02:00  (00:00)
root     pts/0        192.168.123.1    Tue Dec 27 01:59 - down   (00:16)
```

2、只显示五个最近登录的账号：

```
[root@localhost ~]# last -n 5 | awk '{print $1}'
root
reboot
root
root
root
```

awk 工作流程是这样的：读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0 则表示所有域,$1 表示第一个域,$n 表示第 n 个域。默认域分隔符是"空白键" 或 "[tab]键",所以$1 表示登录用户，$3 表示登录用户 ip,以此类推

3、显示/etc/passwd 的账户：

```
[root@localhost ~]# cat /etc/passwd |awk  -F ':'  '{print $1}'
root
bin
daemon
adm
lp
```

这种是 awk+action 的示例，每行都会执行 action{print $1}。
-F 指定域分隔符为':'

4、显示/etc/passwd 的账户和账户对应的 shell,而账户与 shell 之间以 tab 键分割

```
[root@localhost ~]# cat /etc/passwd |awk  -F ':'  '{print $1"\t"$7}'
root    /bin/bash
bin     /sbin/nologin
daemon  /sbin/nologin
adm     /sbin/nologin
lp      /sbin/nologin
```

5、BEGIN and END

如果只是显示/etc/passwd 的账户和账户对应的 shell,而账户与 shell 之间以逗号分割,而且在所有行添加列名 name,shell,在最后一行添加"blue,/bin/nosh"。

cat /etc/passwd |awk -F ':' 'BEGIN {print "name,shell"} {print $1","$7} END {print "blue,/bin/nosh"}'

cat /etc/passwd | awk -F ':' 'BEGIN {print "name \t shell"} {print$1"\t"$7} END {print "blue,/bin/bash"}'

name,shell
root,/bin/bash
daemon,/bin/sh
....
blue,/bin/nosh

awk 工作流程是这样的：先执行 BEGIN，然后读取文件，读入有/n 换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0 则表示所有域,$1 表示第一个域,$n 表示第 n 个域,随后开始执行模式所对应的动作 action。接着开始读入第二条记录•直到所有的记录都读完，最后执行 END 操作。

6、搜索/etc/passwd 有 root 关键字的所有行

awk -F: '/root/' /etc/passwd

root:x:0:0:root:/root:/bin/bash

这种是 pattern 的使用示例，匹配了 pattern(这里是 root)的行才会执行 action(没有指定 action，默认输出每行的内容)。

搜索支持正则，例如找 root 开头的: awk -F: '/^root/' /etc/passwd

搜索/etc/passwd 有 root 关键字的所有行，并显示对应的 shell

awk -F ':' '/root/{print $7}' /etc/passwd

/bin/bash

这里指定了 action{print $7}

6、 awk 常见内置变量

FILENAME：awk 浏览的文件名

FNR：浏览文件的记录数，也就是行数。awk 是以行为单位处理的，所以每行就是一个记录

NR：awk 读取文件每行内容时的行号

NF：浏览记录的域的个数。可以用它来输出最后一个域

FS：设置输入域分隔符，等价于命令行-F 选项

OFS：输出域分隔符

统计/etc/passwd:文件名，每行的行号，每行的列数，对应的完整行内容

awk -F ':' '{print "filename:" FILENAME ",linenumber:" NR ",columns:" NF ",linecontent:"$0}' /etc/passwd

awk -F':' '{print "filename:" FILENAME ",linenumber:" NR ",colums:" NF "linecotent:" $0}' /etc/passwd

filename:/etc/passwd,linenumber:3,columns:7,linecontent:bin:x:2:2:bin:/bin:/bin/sh
filename:/etc/passwd,linenumber:4,columns:7,linecontent:sys:x:3:3:sys:/dev:/bin/sh

使用 printf 替代 print,可以让代码更加简洁，易读

```
awk -F ':' '{printf("filename:%s,linenumber:%s,columns:%s,linecontent:%s\n",FILENAME,NR,NF,$0)}' /etc/passwd
```

指定输入分隔符，指定输出分隔符：

```
awk 'BEGIN {FS=":"; OFS="\t"} {print $1, $2}' /etc/passwd
sshd    x
tcpdump x
linux   x
```

8、实用例子

A：打印最后一列：

```
awk -F: '{print $NF}' /etc/passwd
awk -F: '{printf("%s\n",$NF);}' /etc/passwd
```

B：统计文件行数：

```
awk 'BEGIN {x=0} {x++} END {print x}' /etc/passwd
```

C：打印 9*9 乘法表：

```
awk 'BEGIN{for(n=0;n++<9;){for(i=0;i++<n;)printf i"*"n"="i*n" ";print ""}}'
awk 'BEGIN {for(i=1;i<=9;i++){for(j=1;j<=i;j++){printf i"*"j"="i*j" ";}print ""}}'
awk 'BEGIN {for(i=9;i>=1;i--){for(j=i;j>=1;j--){printf i"*"j"="i*j" ";}print ""}}'
```

D：计算 1-100 之和：

```
echo "sum" | awk 'BEGIN {sum=0;} {i=0;while(i<101){sum+=i;i++}} END {print sum}'
```

9、更多详细用法参见官网：http://www.gnu.org/software/gawk/manual/gawk.html

5.8、find

功能： 搜索文件目录层次结构

格式： find path -option actions

```
find  <路径>  <选项>  [表达式]
```

常用可选项：

-name 根据文件名查找，支持('* ', '? ')

-type 根据文件类型查找(f-普通文件，c-字符设备文件，b-块设备文件，l-链接文件，d-目录)

-perm 根据文件的权限查找，比如 755

-user 根据文件拥有者查找

-group 根据文件所属组寻找文件

-size 根据文件小大寻找文件

-o 表达式 或

-a　　表达式 与
-not　　表达式 非

示例：

```
[linux@linux txt]$ ll     ## 准备的测试文件
total 248
-rw-rw-r--. 1 linux linux 235373 Apr 18 00:10 hw.txt
-rw-rw-r--. 1 linux linux      0 Apr 22 05:43 LINUX.pdf
-rw-rw-r--. 1 linux linux      3 Apr 22 05:50 liujialing.jpg
-rw-rw-r--. 1 linux linux      0 Apr 22 05:43 mingxing.pdf
-rw-rw-r--. 1 linux linux     57 Apr 22 04:40 mingxing.txt
-rw-rw-r--. 1 linux linux     66 Apr 22 05:15 sort.txt
-rw-rw-r--. 1 linux linux    214 Apr 18 10:08 test.txt
-rw-rw-r--. 1 linux linux     24 Apr 22 05:27 uniq.txt

[linux@linux txt]$ find /home/linux/txt/ -name "*.txt"    ## 查找文件名 txt 结尾的文件
/home/linux/txt/uniq.txt
/home/linux/txt/mingxing.txt
/home/linux/txt/test.txt
/home/linux/txt/hw.txt
/home/linux/txt/sort.txt

## 忽略大小写查找文件名包含 linux        -i表示查找时忽略大小写
[linux@linux txt]$ find /home/linux/txt -iname "*linux*"
/home/linux/txt/LINUX.pdf

## 查找文件名结尾是.txt 或者.jpg 的文件   命令中用括号一定要用反斜杠转义，并且括号里的内容和左右括号之间一定要空一个空格
[linux@linux txt]$ find /home/linux/txt/ \( -name "*.txt" -o -name "*.jpg" \)
/home/linux/txt/liujialing.jpg
/home/linux/txt/uniq.txt
/home/linux/txt/mingxing.txt
/home/linux/txt/test.txt
/home/linux/txt/hw.txt
/home/linux/txt/sort.txt
另一种写法：find /home/linux/txt/ -name "*.txt" -o -name "*.jpg"

使用正则表达式的方式去查找上面条件的文件：
[linux@linux txt]$ find /home/linux/txt/ -regex ".*\(\.txt\|\.jpg\)$"
/home/linux/txt/liujialing.jpg
/home/linux/txt/uniq.txt
/home/linux/txt/mingxing.txt
/home/linux/txt/test.txt
/home/linux/txt/hw.txt
/home/linux/txt/sort.txt
```

```
## 查找.jpg 结尾的文件，然后删掉
[linux@linux txt]$ find /home/linux/txt -type f -name "*.jpg" -delete
[linux@linux txt]$ ll
total 248
-rw-rw-r--. 1 linux linux 235373 Apr 18 00:10 hw.txt
-rw-rw-r--. 1 linux linux      0 Apr 22 05:43 LINUX.pdf
-rw-rw-r--. 1 linux linux      0 Apr 22 05:43 mingxing.pdf
-rw-rw-r--. 1 linux linux     57 Apr 22 04:40 mingxing.txt
-rw-rw-r--. 1 linux linux     66 Apr 22 05:15 sort.txt
-rw-rw-r--. 1 linux linux    214 Apr 18 10:08 test.txt
-rw-rw-r--. 1 linux linux     24 Apr 22 05:27 uniq.txt
```

3、Shell 操作字符串

3.1、字符串截取

Linux 中操作字符串，也是一项必备的技能。其中尤以截取字符串更加频繁，下面为大家介绍几种常用方式，截取字符串

1、 #截取，删除左边字符串（包括制定的分隔符），保留右边字符串

预先定义一个变量：WEBSITE='http://hadoop//centos/huangbo.html'

```
[root@hadoop ~]# echo ${WEBSITE#*//}
结果：hadoop//centos/huangbo.html
```

2、 **##截取，删除左边字符串（包括指定的分隔符），保留右边字符串**，和上边一个#不同的是，它一直找到最后，而不是像一个#那样找到一个就满足条件退出了。

```
[root@hadoop ~]# echo ${WEBSITE##*//}
结果：centos/huangbo.html
```

3、 %截取，删除右边字符串（包括制定的分隔符），保留左边字符串

```
[root@hadoop ~]# echo ${WEBSITE%//*}
结果：http://hadoop
```

4、 **%%截取，删除右边字符串（包括指定的分隔符），保留左边字符串**，和上边一个%不同的是，它一直找到最前，而不是像一个%那样找到一个就满足条件退出了。

```
[root@hadoop ~]# echo ${WEBSITE%%//*}
结果：http:
```

总结以上四种方式：

 # 去掉左边，最短匹配模式， ##最长匹配模式。

% 去掉右边，最短匹配模式， %%最长匹配模式

5、 从左边第几个字符开始，以及截取的字符的个数

```
[root@hadoop ~]# echo ${WEBSITE:2:2}
结果：tp
```

6、 从左边第几个字符开始，一直到结束

```
[root@hadoop ~]# echo ${WEBSITE:2}
结果：tp://hadoop//centos//huangbo.html
```

7、 从右边第几个字符开始，以及字符的个数

```
[root@hadoop ~]# echo ${WEBSITE:0-4:2}
结果：ht
```

8、 从右边第几个字符开始，一直到结束

```
[root@hadoop ~]# echo ${WEBSITE:0-4}
结果：html
```

9、 利用 awk 进行字符串截取

```
[root@hadoop ~]# echo $WEBSITE | awk '{print substr($1,2,6)}'
结果：ttp://
```

10、利用 cut 进行字符串截取

```
[root@hadoop ~]# echo $WEBSITE | cut -b 1-4
http
[root@hadoop ~]# echo $WEBSITE | cut -c 1-4
http
[root@hadoop ~]# echo $WEBSITE | cut -b 1,4
hp
[root@hadoop ~]# echo $WEBSITE | cut -c 1,4
hp
```

11、获取最后几个字符

```
[root@hadoop ~]# echo ${WEBSITE:(-3)}
结果：tml
```

12、截取从倒数第 3 个字符后的 2 个字符

```
[root@hadoop ~]# echo ${WEBSITE:(-3):2}
结果：tm
```

3.2、字符串替换

使用格式：${parameter/pattern/string}
例子：

```
定义变量 VAR：
[linux@linux ~]$ VAR="hello tom, hello kitty, hello xiaoming"

替换第一个 hello：
[linux@linux ~]$ echo ${VAR/hello/hi}
hi tom, hello kitty, hello xiaoming

替换所有 hello：
[linux@linux ~]$ echo ${VAR//hello/hi}
hi tom, hi kitty, hi xiaoming
```

3.3、获取字符串长度

在此为大家提供五种方式获取某字符串的长度

1、 使用 wc -L 命令

```
[root@hadoop ~]# echo ${WEBSITE} |wc -L
35
```

2、 使用 expr 的方式去计算

```
[root@hadoop ~]# expr length ${WEBSITE}
35
```

3、 通过 awk + length 的方式获取字符串长度

```
[root@hadoop ~]# echo ${WEBSITE} | awk '{print length($0)}'
35
```

4、 通过 awk 的方式计算以""分隔的字段个数

```
[root@hadoop ~]# echo ${WEBSITE} |awk -F "" '{print NF}'
35
```

5、 通过#的方式获取字符串（最简单，最常用）

```
[root@hadoop ~]# echo ${#WEBSITE}
35
```

4、Shell 脚本自动安装 MySQL

安装 mysql 脚本：

```
#!/bin/bash

## auto install mysql
## 假如是第二次装，那么要先停掉服务，并且卸载之前的 mysql
service mysql stop
EXISTS_RPMS=`rpm -qa | grep -i mysql`
echo ${EXISTS_RPMS}
for RPM in ${EXISTS_RPMS}
do
        rpm -e --nodeps ${RPM}
done

## 删除残留文件
rm -fr /usr/lib/mysql
rm -fr /usr/include/mysql
rm -f /etc/my.cnf
rm -fr /var/lib/mysql

## 从服务器获取安装 mysql 的 rpm 包
wget http://linux/soft/MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm
wget http://linux/soft/MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm

## 删除之前的密码文件，以免产生干扰
rm -rf /root/.mysql_secret

## 安装服务器
rpm -ivh MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm

## 获取到生成的随机密码
##PSWD=`cat /root/.mysql_secret | awk -F ':' '{print substr($4,2,16)}'`
PSWD=` grep -v '^$' /root/.mysql_secret | awk -F ':' '{print substr($4,2,16)}'`
##PSWD=${PWD:1:16}

## 安装客户端
rpm -ivh MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm

## 然后删除刚刚下下来的 rpm 包
rm -rf MySQL-client-5.6.26-1.linux_glibc2.5.x86_64.rpm
rm -rf MySQL-server-5.6.26-1.linux_glibc2.5.x86_64.rpm
```

```
## 提示安装的步骤都完成了。
echo "install mysql server and client is done .!!!!!!"

## 打印出来刚刚生成的 mysql 初始密码
echo "random password is:${PSWD}"

## 开启 mysql 服务
service mysql start
```

手动第一次登陆，然后改掉密码

```
[root@hadoop bin]# mysql -uroot -pZjVIWvOGD18bT7oX
mysql> set PASSWORD=PASSWORD('root');
random password is:ZjVIWvOGD18bT7oX
Starting MySQL.                                            [  OK  ]
[root@hadoop bin]# mysql -uroot -pZjVIWvOGD18bT7oX
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.26

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> set PASSWORD=PASSWORD('root');
Query OK, 0 rows affected (0.00 sec)

mysql>
```

现在就可以写脚本链接 mysql 进行操作了

```
[root@hadoop bin]# vi initMysql.sh


#!/bin/bash


mysql -uroot -proot << EOF
        GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT
OPTION;
        FLUSH PRIVILEGES;
        use mysql;
        select host, user, password from user;
EOF
```

```
[root@hadoop bin]# initMysql.sh
Warning: Using a password on the command line interface can be insecure.
host    user    password
localhost       root    *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B
hadoop  root    *FCACBC1E248D6ABFA89ADB819D4F3667F8A8A1E4
127.0.0.1       root    *FCACBC1E248D6ABFA89ADB819D4F3667F8A8A1E4
::1     root    *FCACBC1E248D6ABFA89ADB819D4F3667F8A8A1E4
%       root    *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B
[root@hadoop bin]#
```