Module01-04
Linux 基础 : find, grep, sed, awk

Linux 基础 - find, grep, sed, awk

- 常用 Linux 命令
- 深入了解 bash
- 正则表达式基础
- ➔ find 、 grep 、 sed 、 awk

Linux 基础 - find 命令

- 常用 Linux 命令
- 深入了解 bash
- 正则表达式基础
- find 、 grep 、 sed 、 awk
  - find
    - grep
    - sed
    - awk
- vi 编辑器

find 命令

- find：查找文件
  - 常用执行方式：
    - find start-dir expression
  - 一些解释：
    - 起始目录：指定 find 开始查找的目录，默认条件下， find 会递归的查找指定目录下所有子目录和文件
    - 表达式 (expression)：表达式通常由 选项、条件测试、动作三部分组成（三者都是可选的），如下面的命令中：

```
$ find ./projects -depth -name '*.sql' -exec cat {} >
sqlfile \;
```

-depth 是选项， -name '*.sql' 是条件测试， -exec cat {} > sqlfile \; 是动作

find 命令 - 表达式

- find 表达式
  - 1，表达式常用选项：
    - -d、 -depth：先找目录下的内容，然后是目录本身
    - -maxdepth <n>：从指定目录最多往下找 n 层
    - -mindepth <n>：从指定目录至少往下找 n 层
    - -prune：防止进入符合某种条件的目录
    - -xdev、 -mount：不要进入其它文件系统查找

find 命令 - 表达式

- find 表达式
  - 2，表达式常用的条件测试：
    - 按文件名查找
      - -name *pattern*、 -iname *pattern*：按指定的文件样式查找， -iname 指不区分大小写

```
# 在当前目录查找以文件名 sql 结尾的文件
$ find . -name '*sql'
$
# 在 /tmp /opt 目录下查找以文件名 sock 结尾的文件
$ find /tmp /opt -name '*sock'
```

    - 按文件类型查找
      - -type filetype：如

```
# 查找 /var 下的 socket 文件
$ find /tmp -type s -ls
```

find 命令 - 表达式

- 按时间查找
  - 说明:
    - 数字 n： n 正好等于 n， -n 小于 n， +n 大于 n
    - a 访问时间， c 状态改变时间， m 修改时间
  - -atime、 -ctime、 -mtime： 单位为日
  - -amin、 -cmin、 -mmin：单位为分钟
  - -anewer file、 -cnewer file、 -newer file：时间比文件 file 新

```
# 查找最近 50 分钟内修改的文件
$ find . -mmin -50
$
# 查找状态更改时间大于 48 小时的文件
$find . -ctime +2
$
# 查找比 samp.sh 文件新的文件
$ find . -newer samp.sh
```

find 命令 - 表达式

- 按文件大小查找
  - 文件的大小单位：
    - b： 以 512 bytes 为单位的块（默认）
    - c： 字节
    - w： 双字节字符数
    - k： 1024 bytes
    - M： 1024*1024 bytes
    - G： 1024*1024*1024 bytes
  - 示例：

```
# 查找文件尺寸大于 10240k 的文件
$ find . -size +10240k
$
# 查找文件尺寸小于 2048 blocks(2048*512 bytes)的文件
$ find . -size -2048
# 查找空文件
$ find . -size 0     # 或 find . -empty
```

- 按权限查询
  - 说明：
    - 权限匹配规则（下面以 220 为例）：
      - 220：文件权限必须为 220
      - -220：文件权限最小为 220，如 550、755 都匹配
      - /220：符合 u,g,o 三个权限中的任意一个皆匹配，如：200,020 等都会匹配，但不匹配 220
    - 数字形式与符号形式的权限表示（举例）：
      - 如：220 → u+w,g+w
      - 如：/650 → /u+rw,g+rx
  - 示例：

```
# 以下 3 条命令一致，查找所有者或组 有写权限的文件
$ find . -perm /220
$ find . -perm /u+w,g+w
$ find . -perm /u=w,g=w
```

find 命令 - 表达式

- 按用户或组查询
  - 说明：
    - 可以是用户名、组名或用户 id 、组 id
    - 查无主的文件： nouser 、 nogroup
    - 查找可读或可写或可执行文件：
      readable 、 writable 、 executable
  - 示例：

```
# 查找 /tmp 下属于用户 kwarph 的文件
$ find /tmp -user kwarph
$ find /tmp -uid 1000        #kwarph 用户的 uid=1000
# 查找 /tmp 下属于用户 kwarph，且属于组 dba 的文件
$ find /tmp -user kwarph -group dba
# 查找当前 /opt 下的无主文件
$ find /opt -nouser -o -nogroup
# 查找当前目录下的可读文件
$ find . -readable
```

怎么会产生无主文件？？

find 命令 - 表达式

- find 表达式
  - 3，表达式常见动作：
    - -print：打印查找到的结果
    - -ls：相当于 ls -l
    - -exec command：执行命令
    - -ok command：同 -exec，但开启交互模式
    - -delete：删除查找到的文件
  - 示例：

```
# 查找当前目录下所有后缀名为 sql 的文件，将其复制到 f2 目录下
$ find . -name '*.sql' -exec cp {} ./f2 \;
# 查找 /opt 下的无主文件，将其删除
$ find /opt -nouser -o -nogroup -ok rm -rf {} \;
# 查找当前目录下所有后缀名为 sql 的文件，并执行 ls -l
$ find . -name '*.sql' -ls
```

find 命令 - 配合 xargs

- 配合 xargs 命令
  - 关于 xargs 命令： xargs 通过管道，将前面命令的输出组装成后一个命令的参数，如：

```
# 将 ls 命令的输出，作为 echo 命令的参数
$ ls *.sql | xargs echo
```

  - 结合 find 与 xargs 的示例

```
# 将当前目录下所有后缀名为 cpp 的文件，打包
$ find . -name '*.cpp' 2>/dev/null | xargs tar cf sql.tar
2>/dev/null
# 相同效果的另外一些做法，见“Shell 语法”单元
$ tar cf sql.tar `find . -name '*.sql' 2>/dev/null`
2>/dev/null
$ tar cf sql.tar $(find . -name '*.sql' 2>/dev/null)
2>/dev/null
```

Linux 基础 - grep 命令

- 常用 Linux 命令
- 深入了解 bash
- 正则表达式基础
- find 、 grep 、 sed 、 awk
  - find
  - grep
  - sed
  - awk
- vi 编辑器

grep 命令

- grep 命令：打印文件中匹配某个样式的行
  - 执行方式： grep [options] pattern [files]
  - 控制样式的选项：
    - `-E`：扩展的 grep， egrep
    - `-f regex-file`：从文件中获取正则表达式， fgrep
    - `-i`：不区分大小写
    - `-v`：只取不匹配的结果
  - 控制输出的选项：
    - `-c`：只输出文件中匹配的行数
    - `-l`：只输出内容匹配的文件名
    - `-L`：与 `-l` 相反，输出内容没有匹配的文件名
    - `-m num`：找到 num 个匹配行就停止继续查找

grep 命令

- 输出行行首格式选项：
  - -n：显示输出行在文件中的行号
  - -h　不输出文件名
  - -H：输出文件名
- 示例：

```
$ cat test_file
just one line here.
Tiger cheng
$ grep -i '^tiger' test_file
Tiger cheng
$ grep -nH '^tiger' test_file
test_file:2:tiger cheng
$
$ ps -ef | grep 'kwarph' | grep -v 'grep'
```

Linux 基础 - sed 命令

- 常用 Linux 命令
- 深入了解 bash
- 正则表达式基础
- find 、 grep 、 sed 、 awk
  - find
  - grep
  - sed
  - awk
- vi 编辑器

sed 命令

- sed：一个用于过滤和转换文字的流编辑器 (Stream Editor)
  - 常规执行方式： sed [options] 'command' [filename]
  - 示例：
    - 单次替换： 's/src/target/'

```
$ echo "tiger's home is home01" | sed 's/home/Home/'
tiger's Home is home01                    # 注意 home01 中的 home 没改变
```

    - 全局替换： 's/src/target/g'

```
$ echo "tiger's home is home01" | sed 's/home/Home/g'
tiger's Home is Home01
```

    - 多次操作： -e 选项（以下 2 种方式效果相同）：

```
$ echo "tiger's home" | sed 's/tiger/Tiger/; s/home/Home/'
$ echo "tiger's home" | sed -e 's/tiger/Tiger/' -e
's/home/Home/'
```

sed 命令

- 示例 (续) :
  - 在匹配样式的行进行替换: '/pattern/ s/src/target'

```
# 原文件内容
$ cat sample_one
one     1
two     1
three   1
one     1
two     1
two     1
three   1
```

```
# 将含有 two 字符串的行中的 1 改为 2
$ sed '/two/ s/1/2/' sample_one
one     1
two     2
three   1
one     1
two     2
two     2
three   1
```

```
# 将含有 two 字符串的行中的 1 改为 2
# 且将含有 three 字符串的行中的 1 改为 3
$ sed '
> /two/ s/1/2/
> /three/ s/1/3/' sample_one
one     1
two     2
three   3       # 以下几行省略
```

- 示例 (续) :
  - 从文件中读取替换规则

```
# 执行上例中的动作: 将含有 two 字符串的行中的 1 改为 2
# 且将含有 three 字符串的行中的 1 改为 3
$ cat replace_list
/two/ s/1/2/
/three/ s/1/3/
$ sed -f replace_list sample_one    # 注意 -f 选项
one     1
two     2
three   3
one     1
two     2
two     2
three   3
```

- 示例 (续) :
  - 指定范围
    - 'n,m s/src/target/'　　(n,m 为行号)
    - '/pattern/,/pattern/ s/src/target/'

```
$ sed '5,6 s/1/2/' sample_one   # 改第 5-6 行中的 1 为 2
one     1
two     1
three   1
one     1
two     2
two     2
three   1
$ sed '/two/,6 s/1/2/' sample_one   # 改第 1 个含 two 的行到第 6 行中的 1 为 2
one     1
two     2
three   2
one     2
......
```

sed 命令

- 示例 (续) :
  - 禁止显示: -n 选项

```
$ sed -n '/two/ s/1/2/' sample_one
                              # 没有输出
$ sed -n '/two/ s/1/2/p' sample_one   # 仅打印匹配的行
two     2
two     2
two     2
$ sed -n '3,5p' sample_one   # 打印 3-5 行
three   1
one     1
two     1
```

sed 命令

- 示例 (续) :
  - 删除行: (d 表示删除, !d 表示不删除)

```
$ sed '/two/ d' sample_one    # 删除行中含 two 的行
one      1
three    1
one      1
three    1
$
$ sed '1,5 d' sample_one      # 删除 1-5 行
two      1
three    1
$ sed '/two/ !d' sample_one   # 不删除包含 two 的行
two      1
two      1
two      1
```

- 示例 (续) :
  - 添加或插入文本:

```
# 在第 3 行后追加一些文本
$ sed '3a\
> extra text' sample_one
one      1
two      1
three    1
extra text
one      1
two      1
two      1
three    1
```

```
# 在第 3 行前插入一些文本
$ sed '3i\
> extra text'  sample_one
one      1
two      1
extra text
three    1
one      1
two      1
two      1
three    1
```

- 示例 (续) :
  - 读写文件:

```
# 将处理后的结果保存到文件 sample_three
$ sed '
> /two/ s/1/2/
> /three/ s/1/3/
> 1,3 w sample_three' sample_one
one     1
two     2
three   3
one     1
two     2
two     2
three   3
$ cat sample_three
one     1
two     2
three   3
```

sed 命令

- 示例 (续) :
  - 修改整行:

```
$ sed '/two/ c\
> We are no longer using two' sample_one
one     1
We are no longer using two
three   1
one     1
We are no longer using two
We are no longer using two
three   1
```

sed 命令

- 示例 (续) :
  - 提前退出:

```
$ sed '
> /two/ s/1/2/
> /three/ s/1/3/
> 5q' sample_one
one     1
two     2
three   3
one     1
two     2
```

```
$ sed '
> /two/ s/1/2/
> /three/ s/1/3/
> /three/q' sample_one
one     1
two     2
three   3
```

```
# 下面命令类似于: head -110 filename
sed 110q filename
```

Linux 基础 - awk 命令

- 常用 Linux 命令
- 深入了解 bash
- 正则表达式基础
- find 、 grep 、 sed 、 awk
  - find
  - grep
  - sed
  - awk
- vi 编辑器

awk 命令

- awk：样式扫描和文本处理语言
  - 执行方式： awk '{pattern + action}' filenames
  - 示例：
    - 示例文件 emp_names 内容

```
$ cat emp_names
46012   DULANEY      EVAN          MOBILE    AL
46013   DURHAM       JEFF          MOBILE    AL
46015   STEEN        BILL          MOBILE    AL
46017   FELDMAN      EVAN          MOBILE    AL
46018   SWIM         STEVE         UNKNOWN   AL
46019   BOGUE        ROBERT        PHOENIX   AZ
46021   JUNE         MICAH         PHOENIX   AZ
46022   KANE         SHERYL        UNKNOWN   AR
46024   WOOD         WILLIAM       MUNCIE    IN
46026   FERGUS       SARAH         MUNCIE    IN
46027   BUCK         SARAH         MUNCIE    IN
46029   TUTTLE       BOB           MUNCIE    IN
```

awk 命令

- 示例（续）：
  - 打印一个或多个字段

```
$ awk '{print $1,$2,$3,$5}' emp_names
46012 DULANEY EVAN AL
46013 DURHAM JEFF AL
46015 STEEN BILL AL
46017 FELDMAN EVAN AL
46018 SWIM STEVE AL
46019 BOGUE ROBERT AZ
46021 JUNE MICAH AZ
46022 KANE SHERYL AR
46024 WOOD WILLIAM IN
46026 FERGUS SARAH IN
46027 BUCK SARAH IN
46029 TUTTLE BOB IN
```

- 示例（续）：
  - 样式匹配

```
$ awk '/AL/ {print $3,$2}' emp_names
EVAN DULANEY
JEFF DURHAM
BILL STEEN
EVAN FELDMAN
STEVE SWIM
$ awk '/AL/' emp_names   # 不指定字段，则打印所有字段
46012   DULANEY      EVAN      MOBILE      AL
46013   DURHAM       JEFF      MOBILE      AL
46015   STEEN        BILL      MOBILE      AL
46017   FELDMAN      EVAN      MOBILE      AL
46018   SWIM         STEVE     UNKNOWN     AL
$ awk '/AL|IN/' emp_names
46012   DULANEY      EVAN      MOBILE     AL
46013   DURHAM       JEFF      MOBILE     AL
......
46029   TUTTLE       BOB       MUNCIE     IN
```

- 示例（续）：
  - 更精确的匹配

```
# 注意结果，在一行中任意字段匹配到 AR
$ awk '/AR/' emp_names
46022   KANE          SHERYL        UNKNOWN   AR
46026   FERGUS        SARAH         MUNCIE    IN
46027   BUCK          SARAH         MUNCIE    IN
$
# 注意格式，现在指定匹配的字段为第五个字段
$ awk '$5~/AR/' emp_names
46022   KANE          SHERYL        UNKNOWN   AR
$ awk '$5!~/AR/' emp_names    # 与上个命令相反
46012   DULANEY       EVAN          MOBILE    AL
46013   DURHAM        JEFF          MOBILE    AL
46015   STEEN         BILL          MOBILE    AL
46017   FELDMAN       EVAN          MOBILE    AL
46018   SWIM          STEVE         UNKNOWN   AL
46019   BOGUE         ROBERT        PHOENIX   AZ
......
```

- 示例（续）：
  - 分行打印：将一行中多个字段分行打印

```
$ awk '/AL/ {print $3,$2 ;
print $4,$5}' emp_names
EVAN DULANEY
MOBILE AL
JEFF DURHAM
MOBILE AL
BILL STEEN
MOBILE AL
EVAN FELDMAN
MOBILE AL
STEVE SWIM
UNKNOWN AL
```

注意右边的结果不符合我们的需求！样式匹配的结果只影响前面的打印语句，而打印 4 ， 5 字段不受影响。

```
$ awk '/AL/ {print $3,$2}
{print $4,$5}' emp_names
EVAN DULANEY
MOBILE AL
JEFF DURHAM
MOBILE AL
BILL STEEN
MOBILE AL
EVAN FELDMAN
MOBILE AL
STEVE SWIM
UNKNOWN AL
PHOENIX AZ
PHOENIX AZ
UNKNOWN AR
MUNCIE IN
MUNCIE IN   # 以下相同的还有 2 行
```

- 示例（续）：
  - 修饰一下输出内容

```
# 字段之间用 tab 隔开
$ awk '{print $1"\t"$2"\t"$3"\t"$4"\t"$5}' emp_names
46012   DULANEY EVAN    MOBILE  AL
46013   DURHAM  JEFF    MOBILE  AL
46015   STEEN   BILL    MOBILE  AL
46017   FELDMAN EVAN    MOBILE  AL
46018   SWIM    STEVE   UNKNOWN AL
46019   BOGUE   ROBERT  PHOENIX AZ
46021   JUNE    MICAH   PHOENIX AZ
46022   KANE    SHERYL  UNKNOWN AR
46024   WOOD    WILLIAM MUNCIE  IN
46026   FERGUS  SARAH   MUNCIE  IN
46027   BUCK    SARAH   MUNCIE  IN
46029   TUTTLE  BOB     MUNCIE  IN
```

- 示例（续）：
  - 花括号：命令组，花括号内为一个命令组，一条 awk 命令可以有多个命令组组成，如

```
$ awk '{print $1, $2} {print $4, $5}' emp_names
```

  - 字段分隔符：
    - 输入字段分隔符： -F 设置，如 /etc/passwd 文件字段用 ':' 分隔。我们可以用 -F":" 指定。指定多个： -F"[:;]"

```
$ awk -F":" '{print $1}' /etc/passwd   或
$ awk '{FS=":"}{print $1}' /etc/passwd
root
daemon
......
```

    - 输出字段分隔符： OFS 设置

```
$ awk '{OFS="-"}{print $1,$2,$3,$4,$5}' emp_names
46012-DULANEY-EVAN-MOBILE-AL      # 其它行略
```

- 示例（续）：
  - 往输出中添加内容

```
$ awk '$5 ~ /IN/ {print "NAME: "$2", "$3"\nCITY-
STATE:"$4", "$5"\n"}' emp_names
NAME: WOOD, WILLIAM
CITY-STATE:MUNCIE, IN

NAME: FERGUS, SARAH
CITY-STATE:MUNCIE, IN

NAME: BUCK, SARAH
CITY-STATE:MUNCIE, IN

NAME: TUTTLE, BOB
CITY-STATE:MUNCIE, IN
```

awk 命令

- awk 算术运算
  - 支持的运算

+	将数字相加
-	减
*	乘
/	除
^	执行指数运算
%	提供模
++	将变量值加一
+=	将其他操作的结果分配给变量
—	将变量减一
-=	将减法操作的结果分配给变量
*=	分配乘法操作的结果
/=	分配除法操作的结果
%=	分配求模操作的结果

- awk 算术运算（续）
  - 示例文件内容

```
$ cat inventory
hammers 5        7.99
drills  2       29.99
punches 7        3.59
drifts  2        4.09
bits   55        1.19
saws  123       14.99
nails 800         .19
screws 80         .29
brads 100         .24
```

- awk 算术运算（续）
  - 算数运算示例

```
# 工具的报表
$ awk '{x=x+($2*$3)}{print $1,"QTY: "$2,"PRICE:
"$3,"TOTAL: "$2*$3,"BAL: "x}' inventory
hammers QTY: 5 PRICE: 7.99 TOTAL: 39.95 BAL: 39.95
drills QTY: 2 PRICE: 29.99 TOTAL: 59.98 BAL: 99.93
punches QTY: 7 PRICE: 3.59 TOTAL: 25.13 BAL: 125.06
drifts QTY: 2 PRICE: 4.09 TOTAL: 8.18 BAL: 133.24
bits QTY: 55 PRICE: 1.19 TOTAL: 65.45 BAL: 198.69
saws QTY: 123 PRICE: 14.99 TOTAL: 1843.77 BAL: 2042.46
nails QTY: 800 PRICE: .19 TOTAL: 152 BAL: 2194.46
screws QTY: 80 PRICE: .29 TOTAL: 23.2 BAL: 2217.66
brads QTY: 100 PRICE: .24 TOTAL: 24 BAL: 2241.66
```

- BEGIN 和 END
  - BEGIN 、 END ： 指定实际处理之前和动作完成之后进行的操作。
  - 示例：

```
$ awk '{x=x+($2*$3)} {print x}' inventory    # 会输出多行
39.95
......
2241.66
$ awk '{x=x+($2*$3)} END {print "Total Value of
Inventory:"x}' inventory          # 只输出一行结果
Total Value of Inventory: 2241.66
$ awk '{x=x+($2*$3)} {print $1,"QTY: "$2,"PRICE:
"$3,"TOTAL: "$2*$3} END {print "Total Value of Inventory: "
x}' inventory
hammers QTY: 5 PRICE: 7.99 TOTAL: 39.95
......
Total Value of Inventory: 2241.66
```

- BEGIN 和 END
  - 示例：

```
$ awk 'BEGIN {print "ITEM\tQUANTITY\tPRICE\tTOTAL"} {x=x+
($2*$3)} {print $1"\t"$2"\t\t"$3"\t"$2*$3} END {print
"Total Value of Inventory: " x}' inventory
ITEM    QUANTITY    PRICE   TOTAL
hammers 5           7.99    39.95
drills  2           29.99   59.98
punches 7           3.59    25.13
drifts  2           4.09    8.18
bits    55          1.19    65.45
saws    123         14.99   1843.77
nails   800         .19     152
screws  80          .29     23.2
brads   100         .24     24
Total Value of Inventory: 2241.66
```

awk 命令

- 输出到文件
  - 示例：

```
$ awk '{print NR, $1 > "/tmp/filez"}' emp_names
$ cat /tmp/filez
1   46012
2   46013
3   46015
4   46017
5   46018
6   46019
7   46021
8   46022
9   46024
10   46026
11   46027
12   46029
```

awk 命令

- 管道
  - 注意：命令必须用双引号括号起来
  - 示例：

```
$ awk '{ print $2 | "sort" }' emp_names
BOGUE
BUCK
DULANEY
DURHAM
FELDMAN
FERGUS
JUNE
KANE
STEEN
SWIM
TUTTLE
WOOD
```

awk 命令

- 调用命令文件
  - 示例：

```
$ cat awklist
{print $3,$2}
{print $4,$5,"\n"}
$ awk -f awklist emp_names
EVAN DULANEY
MOBILE AL

JEFF DURHAM
MOBILE AL

BILL STEEN
MOBILE AL
......
```

find, grep, sed, awk 命令总结

- 在 UNIX 系统中， find 命令的作用远远超过了单纯的文件查找，更多的是配合其它命令完成许多复杂的任务。
- 而 grep 、 sed 、 awk 三个命令，可以完成诸多文本处理的工作。
- 在示例中，对于 grep 、 sed 、 awk 我们很少使用正则表达式，实际情况是，这三个工具是 UNIX 系统中正则表达式功能最为强大的命令。
- sed 、 awk 功能非常丰富，同时也很复杂，在某些时候，澄清一些问题的有效手段是查看它们的 manpages 。