Shell编程

第1章 Shell概述

大数据和全栈工程师程为什么要学习Shell呢?

1）需要看懂运维人员编写的Shell程序。

2）偶尔会编写一些简单Shell程序来管理集群、提高开发效率。

Shell是一个命令行解释器，它接收应用程序/用户命令，然后调用操作系统内核。

Shell还是一个功能相当强大的编程语言，易编写、易调试、灵活性强。

第2章Shell解析器

（1）Linux提供的Shell解析器有：

```
[laoxiao@hadoop101 ~]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/dash
/bin/tcsh
/bin/csh
```

（2）bash和sh的关系

```
[laoxiao@hadoop101 bin]$ ll | grep bash
-rwxr-xr-x. 1 root root 941880 5月  11 2016 bash
lrwxrwxrwx. 1 root root      4 5月  27 2017 sh -> bash
```

（3）Centos默认的解析器是bash

```
[laoxiao@hadoop102 bin]$ echo $SHELL
/bin/bash
```

第3章 Shell脚本入门

1．脚本格式

脚本以#!/bin/bash开头（指定解析器）

2．第一个Shell脚本：helloworld

（1）需求：创建一个Shell脚本，输出helloworld

（2）案例实操：

```
[laoxiao@hadoop101 datas]$ touch helloworld.sh
[laoxiao@hadoop101 datas]$ vi helloworld.sh
```

```
在helloworld.sh中输入如下内容
#!/bin/bash
echo "helloworld"
```

（3）脚本的常用执行方式

第一种：采用bash或sh+脚本的相对路径或绝对路径（不用赋予脚本+x权限）

sh+脚本的相对路径

```
[laoxiao@hadoop101 datas]$ sh helloworld.sh
Helloworld
```

sh+脚本的绝对路径

```
[laoxiao@hadoop101 datas]$ sh /home/laoxiao/datas/helloworld.sh
helloworld
```

bash+脚本的相对路径

```
[laoxiao@hadoop101 datas]$ bash helloworld.sh
Helloworld
```

第二种：采用输入脚本的绝对路径或相对路径执行脚本（**必须具有可执行权限+x**）

（a）首先要赋予helloworld.sh 脚本的+x权限

```
[laoxiao@hadoop101 datas]$ chmod 777 helloworld.sh
```

（b）执行脚本

相对路径

```
[laoxiao@hadoop101 datas]$ ./helloworld.sh
Helloworld
```

注意：第一种执行方法，本质是bash解析器帮你执行脚本，所以脚本本身不需要执行权限。第二种执行方法，本质是脚本需要自己执行，所以需要执行权限。

3．第二个Shell脚本：多命令处理

（1）需求：

在/home/laoxiao/目录下创建一个banzhang.txt,在banzhang.txt文件中增加“I love cls”。

（2）案例实操：

```
[laoxiao@hadoop101 datas]$ touch batch.sh
[laoxiao@hadoop101 datas]$ vi batch.sh
```

在batch.sh中输入如下内容

```
#!/bin/bash


cd /home/laoxiao
touch cls.txt
echo "I love cls" >>cls.txt
```

第4章 Shell中的变量

1、系统（环境）变量

1. **常用系统变量**： 作用域是整个操作系统或者整个用户，注意：临时的用户环境变量只作用到当前bash及它的子bash。

$HOME、$PWD、$SHELL、$USER等

2．案例实操

（1）查看系统变量的值

```
[laoxiao@hadoop101 datas]$ echo $HOME
/home/laoxiao
```

（2）显示当前Shell中所有变量：set

```
[laoxiao@hadoop101 datas]$ set
BASH=/bin/bash
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
```

2、自定义变量

普通变量： 作用域是当前shell（当前的解释器）

1．基本语法

（1）定义变量：变量=值

（2）撤销变量：unset 变量

（3）声明静态变量：readonly变量，注意：不能unset

2．变量定义规则

（1）变量名称可以由字母、数字和下划线组成，但是不能以数字开头，环境变量名建议大写。

（2）等号两侧不能有空格

（3）变量的值如果有空格，需要使用双引号或单引号括起来。

3．案例实操

（1）定义变量A

```
[laoxiao@hadoop101 datas]$ A=5
[laoxiao@hadoop101 datas]$ echo $A
5
```

（2）给变量A重新赋值

```
[laoxiao@hadoop101 datas]$ A=8
[laoxiao@hadoop101 datas]$ echo $A
8
```

（3）撤销变量A

```
[laoxiao@hadoop101 datas]$ unset A
[laoxiao@hadoop101 datas]$ echo $A
```

（4）声明静态的变量B=2，不能unset

```
[laoxiao@hadoop101 datas]$ readonly B=2
[laoxiao@hadoop101 datas]$ echo $B
2
[laoxiao@hadoop101 datas]$ B=9
-bash: B: readonly variable
```

（5）在bash中，变量默认类型都是字符串类型，无法直接进行数值运算

```
[laoxiao@hadoop102 ~]$ C=1+2  # 错误的运算符
[laoxiao@hadoop102 ~]$ echo $C
1+2
```

（6）变量的值如果有空格，需要使用双引号或单引号括起来

```
[laoxiao@hadoop102 ~]$ D=I love banzhang
-bash: world: command not found
[laoxiao@hadoop102 ~]$ D="I love banzhang"
[laoxiao@hadoop102 ~]$ echo $A
I love banzhang
```

（7）可把变量提升为全局环境变量，可供其他Shell程序使用

记住： 用户登录之后开启一个解释器bash（一号），当启动一个脚本文件：重新启动一个bash（二号）去执行脚本，二号bash是一号子bash。

export 变量名

```
[laoxiao@hadoop101 datas]$ vim helloworld.sh
```

在helloworld.sh文件中增加echo $B

```
#!/bin/bash


echo "helloworld"
echo $B


[laoxiao@hadoop101 datas]$ ./helloworld.sh
Helloworld
```

发现并没有打印输出变量B的值。

```
[laoxiao@hadoop101 datas]$ export B
[laoxiao@hadoop101 datas]$ ./helloworld.sh
helloworld
2
```

3、 特殊变量：$n

1．基本语法

$n （功能描述：n为数字，$0代表该脚本名称，$1-$9代表第一到第九个参数，十以上的参数，十以上的参数需要用大括号包含，如${10}）

2．案例实操

（1）输出该脚本文件名称、输入参数1和输入参数2 的值

```
[laoxiao@hadoop101 datas]$ touch parameter.sh
[laoxiao@hadoop101 datas]$ vim parameter.sh

#!/bin/bash

echo "$0  $1   $2"


[laoxiao@hadoop101 datas]$ chmod 777 parameter.sh
[laoxiao@hadoop101 datas]$ ./parameter.sh cls  xz
./parameter.sh  cls   xz
```

4、特殊变量：$#

1．基本语法

$# （功能描述：获取所有输入参数个数，常用于循环）。

2．案例实操

（1）获取输入参数的个数

```
[laoxiao@hadoop101 datas]$ vim parameter.sh


#!/bin/bash
echo "$0  $1   $2"
echo $#


[laoxiao@hadoop101 datas]$ chmod 777 parameter.sh
[laoxiao@hadoop101 datas]$ ./parameter.sh cls  xz
parameter.sh cls xz
2
```

5 、特殊变量：$* 和$@

1．基本语法

$* （功能描述：这个变量代表命令行中所有的参数，$*把所有的参数看成一个整体）

$@ （功能描述：这个变量也代表命令行中所有的参数，不过$@把每个参数区分对待）

2．案例实操

（1）打印输入的所有参数

```
[laoxiao@hadoop101 datas]$ vim parameter.sh


#!/bin/bash

echo "$0  $1   $2"
echo $#
echo $*
echo $@


[laoxiao@hadoop101 datas]$ bash parameter.sh 1 2 3
parameter.sh  1   2
3
1 2 3
1 2 3
```

6、特殊变量：$?

1．基本语法

$? （功能描述：最后一次执行的命令的返回状态。如果这个变量的值为0，证明上一个命令正确执行；如果这个变量的值为非0（具体是哪个数，由命令自己来决定），则证明上一个命令执行不正确了。）

2．案例实操

（1）判断helloworld.sh脚本是否正确执行

```
[laoxiao@hadoop101 datas]$ ./helloworld.sh

hello world

[laoxiao@hadoop101 datas]$ echo $?
0
```

第5章 运算符

1．基本语法

（1）“$((运算式))”或“$[运算式]”

（2）expr + , - , *, /, % 加，减，乘，除，取余

注意：expr运算符间要有空格

2．案例实操：

（1）计算3+2的值

```
[laoxiao@hadoop101 datas]$ expr 2 + 3
5
```

（2）计算3-2的值

```
[laoxiao@hadoop101 datas]$ expr 3 - 2
1
```

（3）计算（2+3）X4的值

（a）expr一步完成计算

```
[laoxiao@hadoop101 datas]$ expr `expr 2 + 3` \* 4
20
```

（b）采用$[运算式]方式

```
[laoxiao@hadoop101 datas]# S=$[(2+3)*4]
[laoxiao@hadoop101 datas]# echo $S
```

第6章 条件判断

1．基本语法

[condition]（**注意condition前后要有空格**）

注意：条件非空即为true，[laoxiao]返回true，[] 返回false。

2. 常用判断条件

（1）两个整数之间比较

= 字符串比较

-lt 小于（less than） -le 小于等于（less equal）

-eq 等于（equal） -gt 大于（greater than）

-ge 大于等于（greater equal） -ne 不等于（Not equal）

（2）按照文件权限进行判断

-r 有读的权限（read） -w 有写的权限（write）

-x 有执行的权限（execute）

（3）按照文件类型进行判断

-f 文件存在并且是一个常规的文件（file）

-e 文件存在（existence） -d 文件存在并是一个目录（directory）

注意：任何命令的执行状态为0则为True，否则为False

3．案例实操

（1）23是否大于等于22

```
[laoxiao@hadoop101 datas]$ [ 23 -ge 22 ]
[laoxiao@hadoop101 datas]$ echo $?
0
```

（2）helloworld.sh是否具有写权限

```
[laoxiao@hadoop101 datas]$ [ -w helloworld.sh ]
[laoxiao@hadoop101 datas]$ echo $?
0
```

（3）/home/laoxiao/cls.txt目录中的文件是否存在

```
[laoxiao@hadoop101 datas]$ [ -e /home/laoxiao/cls.txt ]
[laoxiao@hadoop101 datas]$ echo $?

1
```

（4）多条件判断（&& 表示前一条命令执行成功时，才执行后一条命令，|| 表示上一条命令执行失败后，才执行下一条命令）

```
[laoxiao@hadoop101 ~]$ [ condition ] && echo OK || echo notok
OK
[laoxiao@hadoop101 datas]$ [ condition ] && [ ] || echo notok
notok
```

第7章 流程控制（重点）

1、 if 判断

1．基本语法

```
if [ 条件判断式 ];then

  程序
```

```
fi

或者

if [ 条件判断式 ]

  then

    程序

fi
```

注意事项：

（1）[条件判断式]，中括号和条件判断式之间必须有空格

（2）if后要有空格

2．案例实操

（1）输入一个数字，如果是1，则输出banzhang zhen shuai，如果是2，则输出cls zhen mei，如果是其它，什么也不输出。

```
[laoxiao@hadoop101 datas]$ touch if.sh
[laoxiao@hadoop101 datas]$ vim if.sh

#!/bin/bash


if [ $1 -eq "1" ]
then
       echo "banzhang zhen shuai"
elif [ $1 -eq "2" ]
then
       echo "cls zhen mei"

fi

[laoxiao@hadoop101 datas]$ chmod 777 if.sh
[laoxiao@hadoop101 datas]$ ./if.sh 1
banzhang zhen shuai
```

2、 case 语句

1．基本语法

```
case $变量名 in

  "值1")

    如果变量的值等于值1，则执行程序1
```

```
    ;;
  "值2")
    如果变量的值等于值2，则执行程序2
    ;;
  …省略其他分支…
  *)
    如果变量的值都不是以上的值，则执行此程序
    ;;
esac
```

注意事项：

1) case行尾必须为单词“in”，每一个模式匹配必须以右括号“）”结束。

2) 双分号“**;;**”表示命令序列结束，相当于java中的break。

3) 最后的“*）”表示默认模式，相当于java中的default。

2．案例实操

（1）输入一个数字，如果是1，则输出banzhang，如果是2，则输出cls，如果是其它，输出renyao。

```
[laoxiao@hadoop101 datas]$ touch case.sh
[laoxiao@hadoop101 datas]$ vim case.sh

!/bin/bash

case $1 in
"1")

        echo "banzhang"

;;

"2")
       echo "cls"

;;

*)

        echo "renyao"
;;
esac


[laoxiao@hadoop101 datas]$ chmod 777 case.sh
[laoxiao@hadoop101 datas]$ ./case.sh 1
1
```

3、 for 循环

1．基本语法（一）

for ((初始值;循环控制条件;变量变化))

do

程序

done

2．案例实操

（1）从1加到100

```
[laoxiao@hadoop101 datas]$ touch for1.sh
[laoxiao@hadoop101 datas]$ vim for1.sh

#!/bin/bash


s=0
for((i=0;i<=100;i++))
do

        s=$[$s+$i]

done
echo $s

[laoxiao@hadoop101 datas]$ chmod 777 for1.sh
[laoxiao@hadoop101 datas]$ ./for1.sh
“5050”
```

3．基本语法（二）

for 变量 in 值1 值2 值3...

do

程序

done

4．案例实操

（1）打印所有输入参数

```
[laoxiao@hadoop101 datas]$ touch for2.sh

[laoxiao@hadoop101 datas]$ vim for2.sh
#!/bin/bash
```

```
#打印数字

for i in $*
    do

      echo "ban zhang love $i "

    done

[laoxiao@hadoop101 datas]$ chmod 777 for2.sh
[laoxiao@hadoop101 datas]$ bash for2.sh cls xz bd
ban zhang love cls
ban zhang love xz
ban zhang love bd
```

（2）比较$*和$@区别

（a）$*和$@都表示传递给函数或脚本的所有参数，不被双引号“”包含时，都以$1 $2 …$n的形式输出所有参数。

```
[laoxiao@hadoop101 datas]$ vim for.sh

#!/bin/bash

for i in $*do

 echo "ban zhang love $i "done for j in $@

do

 echo "ban zhang love $j"

done

 [laoxiao@hadoop101 datas]$ bash for.sh cls xz bdban zhang love cls ban zhang love xz
ban zhang love bd ban zhang love clsban zhang love xzban zhang love bd
```

（b）当它们被双引号“”包含时，“$*”会将所有的参数作为一个整体，以“$1 $2 …$n”的形式输出所有参数；“$@”会将各个参数分开，以“$1” “$2”…”$n”的形式输出所有参数。

```
[laoxiao@hadoop101 datas]$ vim for.sh


#!/bin/bash



for i in "$*"

#$*中的所有参数看成是一个整体，所以这个for循环只会循环一次

        do
```

```
                echo "ban zhang love $i"

        done

for j in "$@"

#$@中的每个参数都看成是独立的，所以“$@”中有几个参数，就会循环几次

         do

                 echo "ban zhang love $j"

done


[laoxiao@hadoop101 datas]$ chmod 777 for.sh
[laoxiao@hadoop101 datas]$ bash for.sh cls xz bd
ban zhang love cls xz bd
ban zhang love cls
ban zhang love xz
ban zhang love bd
```

4、 while 循环

1．基本语法

while [条件判断式]

do

程序

done

2．案例实操

（1）从1加到100

```
[laoxiao@hadoop101 datas]$ vim while.sh

#!/bin/bash
s=0
i=1
while [ $i -le 100 ]
do

        s=$[$s+$i]

        i=$[$i+1]

done
echo $s
```

```
[laoxiao@hadoop101 datas]$ chmod 777 while.sh
[laoxiao@hadoop101 datas]$ ./while.sh
5050
```

第8章 read读取控制台输入

1．基本语法

read(选项)(参数)

选项：

-p：指定读取值时的提示符；

-t：指定读取值时等待的时间（秒）。

参数

变量：指定读取值的变量名

2．案例实操

（1）提示7秒内，读取控制台输入的名称

```
[laoxiao@hadoop101 datas]$ touch read.sh
[laoxiao@hadoop101 datas]$ vim read.sh

#!/bin/bash

read -t 7 -p "Enter your name in 7 seconds " NAME
echo $NAME

[laoxiao@hadoop101 datas]$ ./read.sh
Enter your name in 7 seconds xiaoze
xiaoze
```

第9章 函数

1、系统函数

1．basename基本语法

basename [string / pathname][suffix] （功能描述：basename命令会删掉所有的前缀包括最后一个（‘/’）字符，然后将字符串显示出来。

选项：

suffix为后缀，如果suffix被指定了，basename会将pathname或string中的suffix去掉。

2．案例实操

（1）截取该/home/laoxiao/banzhang.txt路径的文件名称

```
[laoxiao@hadoop101 datas]$ basename /home/laoxiao/banzhang.txt
banzhang.txt
[laoxiao@hadoop101 datas]$ basename /home/laoxiao/banzhang.txt .txt
banzhang
```

3、dirname基本语法

dirname 文件绝对路径 （功能描述：从给定的包含绝对路径的文件名中去除文件名（非目录的部分），然后返回剩下的路径（目录的部分））

4．案例实操

（1）获取banzhang.txt文件的路径

```
[laoxiao@hadoop101 ~]$ dirname /home/laoxiao/banzhang.txt
/home/laoxiao
```

2、 自定义函数

1．基本语法

[function] funname[()]

{

Action;

[return int;]

}

funname

2．经验技巧

（1）必须在调用函数地方之前，先声明函数，shell脚本是逐行运行。不会像其它语言一样先编译。

（2）函数返回值，只能通过$?系统变量获得，可以显示加：return返回，如果不加，将以最后一条命令运行结果，作为返回值。return后跟数值n(0-255)

（3）可以采用标准输出：echo来作为函数的返回值

3．案例实操

（1）计算两个输入参数的和

```
[laoxiao@hadoop101 datas]$ touch fun.sh
[laoxiao@hadoop101 datas]$ vim fun.sh

#!/bin/bash
```

```
function sum()
{

    s=0

    s=$[ $1 + $2 ]

    echo "$s"

}


read -p "Please input the number1: " n1;
read -p "Please input the number2: " n2;
sum $n1 $n2;

[laoxiao@hadoop101 datas]$ chmod 777 fun.sh
[laoxiao@hadoop101 datas]$ ./fun.sh
Please input the number1: 2
Please input the number2: 5
7
```

第10章 Shell工具（重点）

1、 cut

cut的工作就是“剪”，具体的说就是在文件中负责剪切数据用的。cut 命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段输出。

1.基本用法

cut [选项参数] filename

说明：默认分隔符是制表符

2.选项参数说明

表1-55

选项参数	功能
-f	列号，提取第几列
-d	分隔符，按照指定分隔符分割列

3.案例实操

（0）数据准备

```
[laoxiao@hadoop101 datas]$ touch cut.txt
[laoxiao@hadoop101 datas]$ vim cut.txt
dong shen
guan zhen
wo  wo
lai  lai
le  le
```

（1）切割cut.txt第一列

```
[laoxiao@hadoop101 datas]$ cut -d " " -f 1 cut.txt
dong
guan
wo
lai
le
```

（2）切割cut.txt第二、三列

```
[laoxiao@hadoop101 datas]$ cut -d " " -f 2,3 cut.txt
shen
zhen
 wo
 lai
 le
```

（3）在cut.txt文件中切割出guan

```
[laoxiao@hadoop101 datas]$ cat cut.txt | grep "guan" | cut -d " " -f 1
guan
```

（4）选取系统PATH变量值，第2个“:”开始后的所有路径：

```
[laoxiao@hadoop101 datas]$ echo $PATH
/usr/lib64/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/laoxi
ao/bin

[laoxiao@hadoop102 datas]$ echo $PATH | cut -d: -f 2-
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/laoxiao/bin
```

2、 sed

sed是一种流编辑器，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。

1. 基本用法

sed [选项参数] ‘command’ filename

2. 选项参数说明

表1-56

选项参数	功能
-e	直接在指令列模式上进行sed的动作编辑。

3. 命令功能描述

表1-57

命令	功能描述
a	新增，a的后面可以接字串，在下一行出现
d	删除
s	查找并替换

4. 案例实操

（0）数据准备

```
[laoxiao@hadoop102 datas]$ touch sed.txt
[laoxiao@hadoop102 datas]$ vim sed.txt
dong shen
guan zhen
wo  wo
lai  lai
le  le
```

（1）将“mei nv”这个单词插入到sed.txt第二行下，打印。

```
[laoxiao@hadoop102 datas]$ sed '2a mei nv' sed.txt
dong shen
guan zhen
mei nv
wo  wo
lai  lai
le  le
```

```
[laoxiao@hadoop102 datas]$ cat sed.txt
dong shen
guan zhen
wo  wo
lai  lai
le  le
```

注意：文件并没有改变， -i：直接修改读取的档案内容

（2）删除sed.txt文件所有包含wo的行

```
[laoxiao@hadoop102 datas]$ sed '/wo/d' sed.txt
dong shen
guan zhen
lai  lai
le  le
```

（3）将sed.txt文件中wo替换为ni

```
[laoxiao@hadoop102 datas]$ sed 's/wo/ni/g' sed.txt
dong shen
guan zhen
ni  ni
lai  lai
le  le
```

注意：‘g’表示global，全部替换

（4）将sed.txt文件中的第二行删除并将wo替换为ni

```
[laoxiao@hadoop102 datas]$ sed -e '2d' -e 's/wo/ni/g' sed.txt
dong shen
ni  ni
lai  lai
le  le
```

3、 awk

一个强大的文本分析工具，把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行分析处理。

1. 基本用法

awk [选项参数] ‘pattern1{action1} pattern2{action2}…’ filename

pattern：表示AWK在数据中查找的内容，就是匹配模式

action：在找到匹配内容时所执行的一系列命令

2. 选项参数说明

表1-55

选项参数	功能
-F	指定输入文件折分隔符
-v	赋值一个用户定义变量

3. 案例实操

（0）数据准备

[laoxiao@hadoop102 datas]$ sudo cp /etc/passwd ./

（1）搜索passwd文件以root关键字开头的所有行，并输出该行的第7列。

```
[laoxiao@hadoop102 datas]$ awk -F: '/^root/{print $7}' passwd
/bin/bash
```

（2）搜索passwd文件以root关键字开头的所有行，并输出该行的第1列和第7列，中间以“，”号分割。

```
[laoxiao@hadoop102 datas]$ awk -F: '/^root/{print $1","$7}' passwd
root,/bin/bash
```

注意：只有匹配了pattern的行才会执行action

（3）只显示/etc/passwd的第一列和第七列，以逗号分割，且在所有行前面添加列名user，shell在最后一行添加"dahaige，/bin/zuishuai"。

```
[laoxiao@hadoop102 datas]$ awk -F : 'BEGIN{print "user, shell"} {print $1","$7}
END{print "dahaige,/bin/zuishuai"}' passwd
user, shell
root,/bin/bash
bin,/sbin/nologin
。。。
laoxiao,/bin/bash
dahaige,/bin/zuishuai
```

注意：BEGIN 在所有数据读取行之前执行；END 在所有数据执行之后执行。

（4）将passwd文件中的用户id增加数值1并输出

```
[laoxiao@hadoop102 datas]$ awk -v i=1 -F: '{print $3+i}' passwd
1
2
3
4
```

4. awk的内置变量

表1-56

变量	说明
FILENAME	文件名
NR	已读的记录数
NF	浏览记录的域的个数（切割后，列的个数）

5. 案例实操

（1）统计passwd文件名，每行的行号，每行的列数

```
[laoxiao@hadoop102 datas]$ awk -F: '{print "filename:"  FILENAME ", linenumber:" NR
",columns:" NF}' passwd
filename:passwd, linenumber:1,columns:7
filename:passwd, linenumber:2,columns:7
filename:passwd, linenumber:3,columns:7
```

（2）切割IP

```
[laoxiao@hadoop102 datas]$ ifconfig ens33 | awk '/inet / {print $2}'

192.168.1.102
```

（3）查询sed.txt中空行所在的行号

```
[laoxiao@hadoop102 datas]$ awk '/^$/{print NR}' sed.txt

5
```

4、 sort

sort命令是在Linux里非常有用，它将文件进行排序，并将排序结果标准输出。

1. 基本语法

sort(选项)(参数)

表1-57

选项	说明
-n	依照数值的大小排序
-r	以相反的顺序来排序
-t	设置排序时所用的分隔字符
-k	指定需要排序的列

参数：指定待排序的文件列表

2. 案例实操

（0）数据准备

```
[laoxiao@hadoop102 datas]$ touch sort.sh
[laoxiao@hadoop102 datas]$ vim sort.sh
bb:40:5.4
bd:20:4.2
xz:50:2.3
cls:10:3.5
ss:30:1.6
```

（1）按照“：”分割后的第三列倒序排序。

```
[laoxiao@hadoop102 datas]$ sort -t : -nrk 3  sort.sh
bb:40:5.4
bd:20:4.2
cls:10:3.5
xz:50:2.3
ss:30:1.6
```

第11章 综合案例（重点）

问题1：使用Linux命令查询file1中空行所在的行号

答案：

```
[laoxiao@hadoop102 datas]$ awk '/^$/{print NR}' sed.txt

5
```

问题2：有文件chengji.txt内容如下：

张三 40

李四 50

王五 60

使用Linux命令计算第二列的和并输出

```
[laoxiao@hadoop102 datas]$ cat chengji.txt | awk -F " " '{sum+=$2} END{print sum}'

150
```

问题3：Shell脚本里如何检查一个文件是否存在？如果不存在该如何处理？

```
#!/bin/bash

if [ -f file.txt ]; then
   echo "文件存在!"

else
   echo "文件不存在!"

fi
```

问题4：用shell写一个脚本，对文本中无序的一列数字排序

```
[root@CentOS6-2 ~]# cat test.txt
9
8
7
6
5
4
3
2
10
1
[root@CentOS6-2 ~]# sort -n test.txt|awk '{a+=$0;print $0}END{print "SUM="a}'
1
2
3
4
5
6
7
8
9
10

SUM=55
```

问题5：请用shell脚本写出查找当前文件夹（/home）下所有的文本文件内容中包含有字符”shen”的文件名称

```
[laoxiao@hadoop102 datas]$ grep -r "shen" /home | cut -d ":" -f 1
/home/laoxiao/datas/sed.txt
/home/laoxiao/datas/cut.txt
```