尚硅谷大数据技术之 Shell

(作者：尚硅谷大数据研发部)

版本：V2.0

第 1 章 Shell 概述

Shell概述

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：

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

（2）bash和sh的关系

```
[atguigu@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

```
[atguigu@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：

```
[atguigu@hadoop101 datas]$ touch helloworld.sh
[atguigu@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

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

sh+脚本的绝对路径

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

bash+脚本的相对路径

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

bash+脚本的绝对路径

```
[atguigu@hadoop101 datas]$ bash /home/atguigu/datas/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

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

（b）执行脚本

相对路径

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

绝对路径

```
[atguigu@hadoop101 datas]$ /home/atguigu/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/atguigu/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：

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

在 batch.sh 中输入如下内容
#!/bin/bash

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

第 4 章 Shell 中的变量

4.1 系统变量

1．常用系统变量

$HOME、$PWD、$SHELL、$USER 等

2．案例实操

（1）查看系统变量的值

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

（2）显示当前 Shell 中所有变量：set

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

4.2 自定义变量

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）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。

（4）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

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

（2）给变量 A 重新赋值

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

（3）撤销变量A

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

（4）声明静态的变量B=2，不能unset

```
[atguigu@hadoop101 datas]$ readonly B=2
[atguigu@hadoop101 datas]$ echo $B
2
[atguigu@hadoop101 datas]$ 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

```
[atguigu@hadoop102 ~]$ C=1+2
[atguigu@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

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

export 变量名

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

在 helloworld.sh 文件中增加 echo $B
#!/bin/bash

echo "helloworld"
echo $B

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

发现并没有打印输出变量B的值。

```
[atguigu@hadoop101 datas]$ export B
```

```
[atguigu@hadoop101 datas]$ ./helloworld.sh
helloworld
2
```

4.3 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

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

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

[atguigu@hadoop101 datas]$ chmod 777 parameter.sh

[atguigu@hadoop101 datas]$ ./parameter.sh cls  xz
./parameter.sh  cls   xz
```

4.4 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

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

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

[atguigu@hadoop101 datas]$ chmod 777 parameter.sh

[atguigu@hadoop101 datas]$ ./parameter.sh cls  xz
parameter.sh cls xz
2
```

4.5 特殊变量：$*、$@

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

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

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

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

4.6 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

```
[atguigu@hadoop101 datas]$ ./helloworld.sh
hello world
[atguigu@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

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

（2）计算 3-2 的值

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

（3）计算（2+3）X4 的值

（a）expr 一步完成计算

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

（b）采用$[运算式]方式

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

3. 案例实操

（1）23 是否大于等于 22

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

（2）helloworld.sh 是否具有写权限

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

（3）/home/atguigu/cls.txt 目录中的文件是否存在

```
[atguigu@hadoop101 datas]$ [ -e /home/atguigu/cls.txt ]
[atguigu@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）

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

第 7 章 流程控制（重点）

7.1 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

elif [条件判断式]

then

程序

else

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。

```
[atguigu@hadoop101 datas]$ touch if.sh
[atguigu@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

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

7.2 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。

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

!/bin/bash
```

```
case $1 in
"1")
        echo "banzhang"
;;

"2")
        echo "cls"
;;
*)
        echo "renyao"
;;
esac

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

7.3 for 循环

1．基本语法 1

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

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

#!/bin/bash

s=0
for((i=0;i<=100;i++))
do
        s=$[$s+$i]
done
echo $s

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

3．基本语法 2

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

```
[atguigu@hadoop101 datas]$ touch for2.sh
[atguigu@hadoop101 datas]$ vim for2.sh

#!/bin/bash
#打印数字

for i in $*
   do
     echo "ban zhang love $i "
   done

[atguigu@hadoop101 datas]$ chmod 777 for2.sh
[atguigu@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。

```
[atguigu@hadoop101 datas]$ touch for.sh
[atguigu@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

[atguigu@hadoop101 datas]$ bash for.sh cls xz bd
ban zhang love cls
ban zhang love xz
ban zhang love bd
ban zhang love cls
ban zhang love xz
ban 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。

```
[atguigu@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

[atguigu@hadoop101 datas]$ chmod 777 for.sh
[atguigu@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
```

7.4 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

```
[atguigu@hadoop101 datas]$ touch while.sh
[atguigu@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

[atguigu@hadoop101 datas]$ chmod 777 while.sh
[atguigu@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

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

#!/bin/bash

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

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

第 9 章 函数

9.1 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/atguigu/banzhang.txt 路径的文件名称

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

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

9.2 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．案例实操

（1）计算两个输入参数的和

```
[atguigu@hadoop101 datas]$ touch fun.sh
[atguigu@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;

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

第 10 章 Shell 工具（重点）

10.1 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
-c	指定具体的字符

3.案例实操

（0）数据准备

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

（1）切割 cut.txt 第一列

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

（2）切割 cut.txt 第二、三列

```
[atguigu@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

```
[atguigu@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：

```
[atguigu@hadoop101 datas]$ echo $PATH
/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin:/usr/loca
l/sbin:/usr/sbin:/sbin:/home/atguigu/bin

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

（5）切割 ifconfig 后打印的 IP 地址

```
[atguigu@hadoop101 datas]$ ifconfig eth0 | grep "inet addr" |
cut -d: -f 2 | cut -d" " -f1
192.168.1.102
```

10.2 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。
-i	直接编辑文件

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

```
[atguigu@hadoop102 datas]$ touch sed.txt
[atguigu@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。

```
[atguigu@hadoop102 datas]$ sed '2a mei nv' sed.txt
dong shen
guan zhen
```

```
mei nv
wo  wo
lai  lai

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

le  le
```

注意：文件并没有改变

（2）删除 sed.txt 文件所有包含 wo 的行

```
[atguigu@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

```
[atguigu@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

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

le  le
```

10.3 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

```
[atguigu@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。

```
[atguigu@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。

```
[atguigu@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"。

```
[atguigu@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
。。。
atguigu,/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

```
[atguigu@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

```
[atguigu@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

```
[atguigu@hadoop102 datas]$ ifconfig eth0 | grep "inet addr" |
awk -F: '{print $2}' | awk -F " " '{print $1}'
192.168.1.102
```

（3）查询 sed.txt 中空行所在的行号

```
[atguigu@hadoop102 datas]$ awk '/^$/{print NR}' sed.txt
5
```

10.4 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

```
[atguigu@hadoop102 datas]$ touch sort.sh
[atguigu@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。

```
[atguigu@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 企业真实面试题

11.1 京东

问题 1：使用 Linux 命令查询 file1 中空行所在的行号

```
答案：
[atguigu@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

```
[atguigu@hadoop102 datas]$ cat chengji.txt | awk -F " " '{sum+=$2}
END{print sum}'
150
```

11.2 搜狐&和讯网

问题 1：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
```

11.3 新浪

问题 1：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
```

11.3 金和网络

问题 1：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

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