SHELL学习笔记

基础

1、基本命令

shell是一个**命令解释器**，它接收应用程序/用户命令，然后调用操作系统内核

脚本是为了方便的快速开发和运维的。

```
root@ubuntu:~# echo $BASH   # 需要大写
/bin/bash
root@ubuntu:~# echo $bash

root@ubuntu:~# df -h   # 查看系统分区
Filesystem      Size  Used Avail Use% Mounted on
udev            934M     0  934M   0% /dev
tmpfs           192M  2.8M  189M   2% /run
/dev/vda1        40G  5.2G   33G  14% /
tmpfs           956M     0  956M   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           956M     0  956M   0% /sys/fs/cgroup
tmpfs           192M     0  192M   0% /run/user/0

root@ubuntu:~/03Shell# cat /etc/shells    # 查看系统中的解析器
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/bin/rbash
/bin/dash

root@ubuntu:/bin# ll |grep bash
-rwxr-xr-x  1 root root 1113504 Jun  7  2019 bash*
lrwxrwxrwx  1 root root       4 Sep 14  2020 rbash -> bash*   # 软链接


```

```
root@ubuntu:/bin# echo $SHELL  默认的解析器，  需大写
/bin/bash

```

2、写一个输出姓名的shell

```
#!/bin/bash        # 以这个开头（指定解析器）
#FILENAME: 01_test.sh  # 下面三行都是注解
#auto echo NAME
#by author liu 2021
echo "liu"    # 输出语句


root@ubuntu:~/03Shell# chmod o+x ./01_test.sh   # 添加权限
root@ubuntu:~/03Shell# ./01_test.sh   # 执行
liu


root@ubuntu:~/03Shell# chmod o-x 01_test.sh
root@ubuntu:~/03Shell# ll
total 12
drwxr-xr-x 2 root root 4096 Jun 21 10:42 ./
drwx------ 9 root root 4096 Jun 21 10:42 ../
-rw-r--r-- 1 root root   82 Jun 21 10:42 01_test.sh
root@ubuntu:~/03Shell# /bin/bash 01_test.sh # 可以直接用/bin/bash执行，这样就不用加执行权限
liu
```

第一种是脚本需要自己执行，所以需要权限，

第二种是bash解析器帮你执行脚本，所以本身不需要执行权限。

3、定义变量（自定义变量和系统变量）

- 自定义变量（**中间不能有空格**）

- 定义变量： 变量=值（**中间不能有空格**）
- 撤销变量： unset 变量
- 声明静态变量：readonly 变量， 注意： 不能unset

注意：

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。

5、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

```
root@ubuntu:/bin# A=1
root@ubuntu:/bin# echo $A
1
root@ubuntu:/bin# unset A
root@ubuntu:/bin# echo $A

root@ubuntu:/bin# readonly B=1
root@ubuntu:/bin# echo $B
1
root@ubuntu:/bin# unset B
-bash: unset: B: cannot unset: readonly variable



root@ubuntu:/bin# C=1+1
root@ubuntu:/bin# echo C
C
root@ubuntu:/bin# echo $C
1+1


root@ubuntu:/bin# S=i love you

```

```
Command 'love' not found, but can be installed with:

root@ubuntu:/bin# S="i love you"   # 中间有空格的，需要加上引号


```

-

- 系统变量（不需要定义，直接使用）
  - $0 : 当前程序名称
  - $n: 当前程序的第n个参数 n = 1,2,,,,,9，十个以上的参数需要使用大括号如 $ {10}
  - $* : 当前程序的所有参数（不包括程序本身）（把所有参数看成一个整体）
  - $@: 代表命令行中所有参数（不包括程序本身）（把每个参数区分对待）
  - $# : 当前程序的参数个数（不包括程序本身）(**常用于循环**)
  - $?: 命令或程序执行完后的状态，一般返回0表示执行成功, 判断上一条命令是否成功。如果是非0（具体什么数，于命令相关）
  - $UID: 当前用户的ID
  - $PWD: 当前所在的目录
  - $HOME: 自己用户的/home目录
  - $SHELL: 查看默认的解析器
  - $USER: 查看当前的用户

测试脚本

```
#!/bin/bash
#define a var
# by liu

A=3
echo "A = $A"
name="liu"
echo "my name is $name"
```

```
9
10 echo ================
11 echo $0
12 echo $1
13 echo $#
14 echo =======
15
16 echo $UID
17 echo $PWD
18
19 ##########################################################
   #
20 执行过程
21 root@ubuntu:~/03Shell# ./02test.sh
22 A = 3
23 my name is liu
24 ================
25 ./02test.sh
26
27 0
28 =======
29 0
30 /root/03Shell
31 root@ubuntu:~/03Shell# ./02test.sh 1 name 2 age
32 A = 3
33 my name is liu
34 ================
35 ./02test.sh
36 1
37 4
38 =======
39 0
40 /root/03Shell
```

4、运算符

1、基本语法

> $((运算式)) 或 $[运算式]
>
> expr + - * / % 加 减 乘 除 取余
>
> **注意:** expr 运算符间要有空格, 乘是*

```
root@ubuntu:/bin# expr 2 +5  # 语法错误
expr: syntax error

root@ubuntu:/bin# expr 2 + 5
7
root@ubuntu:/bin# expr 2 \* 5
10

root@ubuntu:/bin# expr `expr 2 + 3` \* 2   # 计算 （2 + 3） * 2
10
也可以按照下面来写
root@ubuntu:/bin# s=$[(2+3)*4]
root@ubuntu:/bin# echo $s
20

```

5、条件判断

> [condition] (注意： condition 前后都要有一个空格，否则报错)
>
> 条件非空即为true， []返回false。

> 逻辑运算符解析：

- -f　　判断文件是否存在　　`if [-f  filename]`
- -d　　判断目录是否存在　　`if [-d  dir ]`

- -e 文件是否存在（exist）
- -eq 等于 应用与 ：整数比较（equal)
- -ne 不等于 应用于： 整数比较（not equal)
- -lt 小于 应用于： 整数比较（less than）
- -gt 大于 应用于： 整数比较 （greater than)
- -le 小于等于 应用于：整数比较 (less equal)
- -ge 大于等于 应用于：整数比较 (greater equal)
- -r 是否有读的权限
- -w 是否有写的权限
- -x 是否有执行权限
- -a 双方都成立（and）逻辑表达式 -a 逻辑表达式
- -o 单方成立（or） 逻辑表达式 -o 逻辑表达式
- -z 空字符串

```
判断目录或者文件是否存在
#!/bin/bash
DIR=$PWD/202106

if [ ! -d $DIR ]; then    # 判断目录是否存在
   mkdir -p $DIR
else
   echo "$DIR is exist!"
fi

echo ======================

FILE=$PWD/202106/test.txt

if [ ! -f $FILE ]; then    # 判断文件是否存在
   echo "OK" >> $FILE
else
#   echo "$FILE is exist!!!!!\n"
   cat $FILE
fi

root@ubuntu:~/03Shell# /bin/bash 04test.sh
/root/03Shell/202106 is exist!
```

```
======================
OK
```

```
root@ubuntu:~/03Shell# [ 23 -ge 32 ]   # 23 大于 32 嘛
root@ubuntu:~/03Shell# echo $?        # 返回1，失败 23 小于32
1

root@ubuntu:~/03Shell# [ 23 -le 32 ]
root@ubuntu:~/03Shell# echo $?
0

root@ubuntu:~/03Shell# ll
total 28
drwxr-xr-x 2 root root 4096 Jun 21 14:24 ./
drwx------ 9 root root 4096 Jun 21 14:24 ../
-rw-r--r-- 1 root root   82 Jun 21 10:42 01_test.sh
-rw-r--r-x 1 root root  172 Jun 21 11:01 02test.sh*
-rw-r--r-- 1 root root   97 Jun 21 11:20 03test.sh
-rw-r--r-- 1 root root  254 Jun 21 12:27 04test.sh
-rw-r--r-x 1 root root  206 Jun 21 14:24 05test.sh*
root@ubuntu:~/03Shell# [ -w 05test.sh ] # 判断文件是否有写的权限， 返回0， 表示有
root@ubuntu:~/03Shell# echo $?
0

root@ubuntu:~/03Shell# [ -e /home/liu/test.txt ] #判断这个目录是否存在，返回1， 不存在
root@ubuntu:~/03Shell# echo $?
1

```

- 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。

```
root@ubuntu:~/03Shell# [ condition ] && echo OK || echo "not OK"
OK
root@ubuntu:~/03Shell# [ condition ] && [] || echo "not OK"   ## 中间需要有空格
[]: command not found
not OK
root@ubuntu:~/03Shell# [ condition ] && [ ] || echo "not OK"
not OK

```

6、 if语句

1、if语句

```
if [ 条件判断式子 ]; then       #  条件表达式子两边必须有空格
   语句1
elif [ 条件判断式子2 ]; then     # 可以没有
   语句2
else
   语句3
fi
```

```
root@ubuntu:~/03Shell# cat 03test.sh
#!/bin/bash
NUM1=10
NUM2=20
if(($NUM1 > $NUM2)); then   # 注意格式
   echo "NUM1 big"
else
   echo "NUM2 big"
fi

root@ubuntu:~/03Shell# /bin/bash 03test.sh
NUM2 big

```

根据参数来打印liu de hua 这三个字符串。

```
#!/bin/bash
#by liu

if [ $1 -eq 1 ]; then
    echo "name is liu";
elif [ $1 -eq 2 ]; then
    echo "name is de";
else
    echo "name is hua"
fi
==========================================================
root@ubuntu:~/03Shell# /bin/bash if.sh 2
name is de
root@ubuntu:~/03Shell# /bin/bash if.sh 1
name is liu
root@ubuntu:~/03Shell# /bin/bash if.sh 3
name is hua

```

7、case语句

> case $变量名 in
>
> “值1)” 等于它，执行程序1
>
> ;;
>
> “值2)” 等于它，执行程序2
>
> ;;
>
> “值3)” 等于它，执行程序3
>
> ;;
>
> “*)” 不等于上面的，执行程序4
>
> ;;

```
esac
```

0、case行尾必须为单词 in， 每个模式匹配必须以右括号 “）” 结束；

1、；；相当于break;

2、*) 相当于default

用case根据参数打印liu de hua 这三个字符串。

```
1
2
3 root@ubuntu:~/03Shell# /bin/bash 06test.sh
4 hua
5 root@ubuntu:~/03Shell# /bin/bash 06test.sh 1
6 liu
7 root@ubuntu:~/03Shell# /bin/bash 06test.sh 2
8 de
9
```

8、for循环

语法1：

```
for（（初始值； 循环控制条件；变量变化））
 do
 程序
 done
```

从1加到100

```
1 #!/bin/bash
2 #by liu
3
4 sum=0
```

```
for((i=1;i<=100;i=i+1))
do
   sum=$[ $sum+$i ]
done

echo "sum = $sum"


root@ubuntu:~/03Shell# /bin/bash 07test.sh
sum = 5050
```

语法2

> for 变量 in 值1 值2 值3 ...
>
> do
>
> 程序
>
> done

输出所有参数

```
#!/bin/bash
#by liu
# print all para

for i in $*
   do echo $i;
   done

root@ubuntu:~/03Shell# /bin/bash 08test.sh 1 2 3  4 55
1
2
3
4
55

```

9、while循环

语法

> while [条件判断式子] # 有三个空格
>
> do
>
> 程序
>
> done

1加到100

```
#!/bin/bash
#by liu
# 1 + ... 100

i=1
sum=0
while [ $i -le 100 ]
do
    sum=$[ $sum + $i ]
    i=$[$i + 1]
done
echo "sum = $sum"

root@ubuntu:~/03Shell# /bin/bash while.sh
sum = 5050
```

10、read读取控制台输入

> read(选项)（参数）
>
> 选项：
>
> -p： 指定读取值时的提示符

> -t： 指定读取值时的等待的时间（s）
>
> 参数
>
> 变量： 指定读取值的变量名

操作：在提示10s内，读取控制台输入的名称

```
#!/bin/bash

read -t 10 -p "input name" NAME

echo $NAME
```

函数

1、系统函数

> basename 用法
>
> basename [string/ pathname][suffix] (功能： 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。

实操：

1、截取该路径的文件名称

```
root@ubuntu:~/03Shell# basename /home/liu/data/test.txt  # 不加后缀 直接截取最后一个/ 后面的
test.txt
root@ubuntu:~/03Shell# basename /home/liu/data/test.txt  .txt  # 加了后缀，在将后缀去掉
test
```

> diename 基本语法
>
> dirname 文件绝对路径 （功能：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（目录部分））

实操：

2、获取文件的路径

```
root@ubuntu:~/03Shell# dirname /home/liu/data/test.txt
/home/liu/data
```

2、自定义函数

1、语法

> [function] funcname[()]
>
> {
>
> Action;
>
> [return int;]
>
> }
>
> funcname

2、技巧

- 必须在调用函数地方之前，先声明函数，shell脚本是逐行运行。不会像其他语言一样先编译。
- 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后跟数据n（0-255）

3、实操：写一个函数，计算两个数之和

```
#/bin/bash

function sum()
{
   s=0;
   s=$[$1+$2]  #只有条件变量左右才有空格
   echo $s
}

read -p "input one para:" P1
read -p "input two para:" P2

sum $P1 $P2

root@ubuntu:~/03Shell# /bin/bash 09test.sh
input one para:2
input two para:3
5
```

Shell工具

cut

> cut的工作就是 剪掉， 具体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、用法

> cut [选项参数] filename
>
> 注意： 默认分割符是制表符

2、选项参数说明

> -f 列号， 提取第几列
>
> -d 分隔符，按照指定的分隔符分割列

3、实操：

```
准备数据
root@ubuntu:~/03Shell# cat 10.txt
liu  de
xiao ming
wang jing
zhang san
li si
wu ji

root@ubuntu:~/03Shell# cut -d " " -f 1 10.txt   #用空格作为分隔符， 输出第1列
liu
xiao
wang
zhang
li
wu

root@ubuntu:~/03Shell# cat 10.txt |grep xiao
xiao ming
root@ubuntu:~/03Shell# cat 10.txt |grep xiao | cut -d " " -f 1   #   利用管道
xiao


#选取系统PATH变量值，第2个：开始后的所有路径
root@ubuntu:~/03Shell# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
root@ubuntu:~/03Shell# echo $PATH | cut -d : -f 3-
/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
root@ubuntu:~/03Shell# echo $PATH | cut -d ":" -f 3-    # 也可以加上""
/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin


```

```
32
33 # 切割ifconfig后打印的ip地址
34 root@ubuntu:~/03Shell# ifconfig
35 eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
36         inet 172.17.186.12  netmask 255.255.240.0  broadcast 172.17.191.255
37         inet6 fe80::216:3eff:fe10:e185  prefixlen 64  scopeid 0x20<link>
38         ether 00:16:3e:10:e1:85  txqueuelen 1000  (Ethernet)
39         RX packets 14887072  bytes 3601180607 (3.6 GB)
40         RX errors 0  dropped 0  overruns 0  frame 0
41         TX packets 12916711  bytes 8524216373 (8.5 GB)
42         TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
43
44 lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
45         inet 127.0.0.1  netmask 255.0.0.0
46         inet6 ::1  prefixlen 128  scopeid 0x10<host>
47         loop  txqueuelen 1000  (Local Loopback)
48         RX packets 4729376  bytes 381326315 (381.3 MB)
49         RX errors 0  dropped 0  overruns 0  frame 0
50         TX packets 4729376  bytes 381326315 (381.3 MB)
51         TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
52
53 root@ubuntu:~/03Shell# ifconfig eth0 | grep "inet " | cut -d " " -f 10- | cut -d " " -f
   1
54 172.17.186.12
55 有的ip地址前面是：就是下面的命令
56 ifconfig eth0 | grep "inet addr" | cut -d : -f 2 | cut -d ' ' -f 1
57
```

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，这样不断重读，直到文件末尾。文件内容并没有改变，除非使用重定向存储输出。

1、用法

> sed [选项参数] 'command' filename
>
> 选项参数说明：
>
> -e 直接在指令列模式上进行sed的动作编辑
>
> 命令功能描述：
>
> a 新增， a的后面可接字串，在下一行出现
>
> d 删除
>
> s 查找并替换

2、实操

```
数据准备
root@ubuntu:~/03Shell# cat 11test.txt
xiao ming
liu hua
zhang san
xiao long

wu ji
le le

================================================================================
1、在第2行插入liu de hua
root@ubuntu:~/03Shell# sed "2a liu de hua" 11test.txt    里面的文件并没有改变
xiao ming
liu hua
liu de hua
zhang san
xiao long

wu ji
le le
root@ubuntu:~/03Shell# cat 11test.txt
xiao ming
```

```
liu hua
zhang san
xiao long

wu ji
le le

2、删除11test.txt中所有包含le的行
root@ubuntu:~/03Shell# sed "/le/d" 11test.txt
xiao ming
liu hua
zhang san
xiao long

wu ji

3、将文件中的xiao 替换为 liu
root@ubuntu:~/03Shell# sed "s/xiao/liu/g" 11test.txt  # 这里的g表示global， 全部替换
liu ming
liu hua
zhang san
liu long    # 可以看到已经改了

wu ji
le le

4、将文件中第二行删除并将xiao 改为liu
root@ubuntu:~/03Shell# sed -e "2d" -e "s/xiao/liu/g" 11test.txt
liu ming
zhang san
liu long

wu ji
le le

```

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，切开的部分在进行分析处理。

> awk同sed类似：只不过**sed 擅长取行；awk 命令擅长取列**
>
> 一般是遍历一个文件中的每一行，然后分别对文件的每一列进行处理

1、用法

> awk [选项参数] 'pattern{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

> 选项参数
>
> -F 指定输入文件分隔符
>
> -v 赋值一个用户定义变量

2、实操

```
数据准备
root@ubuntu:~/03Shell# cp /etc/passwd ./
root@ubuntu:~/03Shell# cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

```
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:105:109::/run/uuidd:/usr/sbin/nologin
ntp:x:106:111::/nonexistent:/usr/sbin/nologin
sshd:x:107:65534::/run/sshd:/usr/sbin/nologin
_chrony:x:108:117:Chrony daemon,,,:/var/lib/chrony:/usr/sbin/nologin
liu:x:1000:1000::/home/liu:/bin/sh
mysql:x:109:118:MySQL Server,,,:/nonexistent:/bin/false

1、搜索passwd文件中以root关键字开头所有行，并输出改行的第7列
root@ubuntu:~/03Shell# 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"," 号为分割
root@ubuntu:~/03Shell# awk -F : '/^root/ {print $1 "," $7}' passwd
root,/bin/bash

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"liudehua, /bin/loveyou".


root@ubuntu:~/03Shell# awk  -F : 'BEGIN{print "user, shell"}{print $1
```

```
","$7}END{print "liudehua, bin/loveyou"}' passwd
user, shell
root,/bin/bash
daemon,/usr/sbin/nologin
bin,/usr/sbin/nologin
sys,/usr/sbin/nologin
sync,/bin/sync
games,/usr/sbin/nologin
man,/usr/sbin/nologin
lp,/usr/sbin/nologin
mail,/usr/sbin/nologin
news,/usr/sbin/nologin
uucp,/usr/sbin/nologin
proxy,/usr/sbin/nologin
www-data,/usr/sbin/nologin
backup,/usr/sbin/nologin
list,/usr/sbin/nologin
irc,/usr/sbin/nologin
gnats,/usr/sbin/nologin
nobody,/usr/sbin/nologin
systemd-network,/usr/sbin/nologin
systemd-resolve,/usr/sbin/nologin
syslog,/usr/sbin/nologin
messagebus,/usr/sbin/nologin
_apt,/usr/sbin/nologin
uuidd,/usr/sbin/nologin
ntp,/usr/sbin/nologin
sshd,/usr/sbin/nologin
_chrony,/usr/sbin/nologin
liu,/bin/sh
mysql,/bin/false
liudehua, bin/loveyou
#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
root@ubuntu:~/03Shell# awk -F : -v i=1 '{print $3+i}' passwd

1
```

```
2
3
4
5
6
7
8
9
10
11
14
34
35
39
40
42
65535
101
102
103
104
105
106
107
108
109
1001
110

```

3、awk的 内置变量

> 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）

4、实操

```
1、统计passwd文件名，每行的行号，每行的列数
root@ubuntu:~/03Shell# awk -F : '{print FILENAME ", " NR ", " NF}' passwd
passwd, 1, 7
passwd, 2, 7
passwd, 3, 7
passwd, 4, 7
passwd, 5, 7
passwd, 6, 7
passwd, 7, 7
passwd, 8, 7
passwd, 9, 7
passwd, 10, 7
passwd, 11, 7
passwd, 12, 7
passwd, 13, 7
passwd, 14, 7
passwd, 15, 7
passwd, 16, 7
passwd, 17, 7
passwd, 18, 7
passwd, 19, 7
passwd, 20, 7
passwd, 21, 7
passwd, 22, 7
passwd, 23, 7
passwd, 24, 7
passwd, 25, 7
passwd, 26, 7
passwd, 27, 7
passwd, 28, 7
passwd, 29, 7

2、切割IP
root@ubuntu:~/03Shell# ifconfig eth0 | grep "inet "
        inet 172.17.186.12  netmask 255.255.240.0  broadcast 172.17.191.255
root@ubuntu:~/03Shell# ifconfig eth0 | grep "inet " | awk -F " " '{print $2}'
172.17.186.12

```

```
3、查询11test.txt 中空行所在的行号 （面试题）
root@ubuntu:~/03Shell# awk '/^$/ {print NR}' 11test.txt
5
root@ubuntu:~/03Shell# cat 11test.txt
xiao ming
liu hua
zhang san
xiao long

wu ji
le le

```

注意：

> ^ 匹配字符串开始的位置。
>
> $ 匹配字符串结尾的位置。

sort

> sort 命令是在linux中非常有用的，它将文件进行排序，并将排序结果标准输出

1、语法

> sort（选项） （参数）
>
> -n 依照数值的大小排序
>
> -r 以相反的顺序来排序
>
> -t 设置排序所用的分割字符
>
> -k 指定需要排序的列
>
> 参数： 指定带排序的文件列表

2、实操：

```
0、数据准备
aa:40:5:3
cb:20:4:3
bc:30:3:4
zx:55:1:6
cfv:467:2:3

root@ubuntu:~/03Shell# cat 12test.txt
aa:40:5:3
cb:20:4:3
bc:30:3:4
zx:55:1:6
cfv:467:2:3
root@ubuntu:~/03Shell# sort -t : -nrk 2 12test.txt
cfv:467:2:3
zx:55:1:6
aa:40:5:3
bc:30:3:4
cb:20:4:3
```

shell的一些面试题

1、使用linux命令查询file1中空行所在的行号（京东）

```
awk '/^$/{print NR}' file1
```

2、使用linux命令计算第二列的和并输出（京东）

```
============数据=========================
chengji.txt
zhangsan    40
lisi        50
wangwu      60
=========================================

cat chengji.txt | awk -F " " '{sum+=$2} END{print sum}'
150
```

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？(搜狐)

```
#!/bin/bash
#by liu

if [ -f file.txt ]; then
   echo "file 存在"
else
   echo "file 不存在"
fi
```

4、用shell写一个脚本，对文本中无序的一列数字排序

```
cat test.txt
9
8
7
6
5
4
3
2
10
```

```
1
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
```

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 "liu" 的文件内容(金和网络)

```
grep -r "liu" /home | cut -d ":" -f 1
/home/data/1.txt
/home/data/2.txt
```