

本节课程目标

- 了解shell中的通配符
- 熟悉grep、cut、sort等小工具和shell中的通配符的使用

#一、文本处理工具

##1. ==grep工具==

grep是==行==过滤工具；用于根据关键字进行行过滤

语法和选项

语法：

```
# grep [选项] '关键字' 文件名
```

常见选项：

OPTIONS：

- i：不区分大小写
- v：查找不包含指定内容的行,反向选择
- w：按单词搜索
- o：打印匹配关键字
- c：统计匹配到的行数
- n：显示行号
- r：逐层遍历目录查找
- A：显示匹配行及后面多少行
- B：显示匹配行及前面多少行
- C：显示匹配行前后多少行
- l：只列出匹配的文件名
- L：列出不匹配的文件名
- e：使用正则匹配
- E：使用扩展正则匹配
- ^key：以关键字开头
- key\$：以关键字结尾
- ^\$：匹配空行
- color=auto：可以将找到的关键词部分加上颜色的显示

颜色显示（别名设置）：

临时设置:

```
# alias grep='grep --color=auto' //只针对当前终端和当前用户生效
```

永久设置:

1) 全局 (针对所有用户生效)

```
vim /etc/bashrc
alias grep='grep --color=auto'
source /etc/bashrc
```

2) 局部 (针对具体的某个用户)

```
vim ~/.bashrc
alias grep='grep --color=auto'
source ~/.bashrc
```

举例说明:

==说明: 不要直接使用/etc/passwd文件, 将其拷贝到/tmp下做实验! ==

# grep -i root passwd	忽略大小写匹配包含root的行
# grep -w ftp passwd	精确匹配ftp单词
# grep -w hello passwd	精确匹配hello单词; 自己添加包含hello的行到文件
# grep -wo ftp passwd	打印匹配到的关键字ftp
# grep -n root passwd	打印匹配到root关键字的行号
# grep -ni root passwd	忽略大小写匹配统计包含关键字root的行
# grep -nic root passwd	忽略大小写匹配统计包含关键字root的行数
# grep -i ^root passwd	忽略大小写匹配以root开头的行
# grep bash\$ passwd	匹配以bash结尾的行
# grep -n ^\$ passwd	匹配空行并打印行号
# grep ^# /etc/vsftpd/vsftpd.conf	匹配以#号开头的行
# grep -v ^# /etc/vsftpd/vsftpd.conf	匹配不以#号开头的行
# grep -A 5 mail passwd	匹配包含mail关键字及其后5行
# grep -B 5 mail passwd	匹配包含mail关键字及其前5行
# grep -C 5 mail passwd	匹配包含mail关键字及其前后5行

##2. ==cut工具==

cut是==列==截取工具, 用于列的截取

语法和选项

语法:

```
# cut 选项 文件名
```

常见选项:

- c: 以字符为单位进行分割, 截取
- d: 自定义分隔符, 默认为制表符\t
- f: 与-d一起使用, 指定截取哪个区域

举例说明:

# cut -d: -f1 1.txt	以:冒号分割, 截取第1列内容
# cut -d: -f1,6,7 1.txt	以:冒号分割, 截取第1,6,7列内容
# cut -c4 1.txt	截取文件中每行第4个字符
# cut -c1-4 1.txt	截取文件中每行的1-4个字符
# cut -c4-10 1.txt	截取文件中每行的4-10个字符
# cut -c5- 1.txt	从第5个字符开始截取后面所有字符

课堂练习:

用小工具列出你当系统的运行级别。5/3

1. 如何查看系统运行级别
 - 命令 `runlevel`
 - 文件 `/etc/inittab`
2. 如何过滤运行级别

```
runlevel | cut -c3
runlevel | cut -d ' ' -f2
grep -v '^#' /etc/inittab | cut -d: -f2
grep '^id' /etc/inittab | cut -d: -f2
grep "initdefault:$" /etc/inittab | cut -c4
grep -v ^# /etc/inittab | cut -c4
grep 'id:' /etc/inittab | cut -d: -f2
cut -d': ' -f2 /etc/inittab | grep -v ^#
cut -c4 /etc/inittab | tail -1
cut -d: -f2 /etc/inittab | tail -1
```

3. sort工具

sort工具用于排序;它将文件的每一行作为一个单位,从首字符向后,依次按ASCII码值进行比较,最后将他们按升序输出。

语法和选项

- u : 去除重复行
- r : 降序排列,默认是升序
- o : 将排序结果输出到文件中,类似重定向符号>
- n : 以数字排序,默认是按字符排序
- t : 分隔符
- k : 第N列
- b : 忽略前导空格。
- R : 随机排序,每次运行的结果均不同

举例说明

# sort -n -t: -k3 1.txt	按照用户的uid进行升序排列
# sort -nr -t: -k3 1.txt	按照用户的uid进行降序排列
# sort -n 2.txt	按照数字排序
# sort -nu 2.txt	按照数字排序并且去重
# sort -nr 2.txt	
# sort -nru 2.txt	
# sort -nru 2.txt	
# sort -n 2.txt -o 3.txt	按照数字排序并将结果重定向到文件
# sort -R 2.txt	
# sort -u 2.txt	

##4.uniq工具

uniq用于去除==连续==的==重复==行

常见选项:

-i: 忽略大小写
-c: 统计重复行次数
-d: 只显示重复行

举例说明:

```
# uniq 2.txt
# uniq -d 2.txt
# uniq -dc 2.txt
```

5.tee工具

tee工具是从标准输入读取并写入到标准输出和文件，即：双向覆盖重定向（屏幕输出|文本输入）

选项:

-a 双向追加重定向

```
# echo hello world
# echo hello world|tee file1
# cat file1
# echo 999|tee -a file1
# cat file1
```

6.diff工具

diff工具用于逐行比较文件的不同

注意：diff描述两个文件不同的方式是告诉我们==怎样改变第一个==文件之后==与第二个文件匹配==。

语法和选项

语法:

```
diff [选项] 文件1 文件2
```

常用选项:

选项	含义	备注
-b	不检查空格	
-B	不检查空白行	
-i	不检查大小写	
-w	忽略所有的空格	
--normal	正常格式显示(默认)	
-c	上下文格式显示	
-u	合并格式显示	

举例说明:

- 比较两个==普通文件==异同，文件准备：

```
[root@MissHou ~]# cat file1
aaaa
111
hello world
222
333
bbb
[root@MissHou ~]#
[root@MissHou ~]# cat file2
aaa
hello
111
222
bbb
333
world
```

1) 正常显示

```
diff目的: file1如何改变才能和file2匹配
[root@MissHou ~]# diff file1 file2
1c1,2                第一个文件的第1行需要改变(c=change)才能和第二个文件的第1到2行匹配
< aaaa              小于号"<"表示左边文件(file1)文件内容
---                ---表示分隔符
> aaa               大于号">"表示右边文件(file2)文件内容
> hello
3d3                第一个文件的第3行删除(d=delete)后能和第二个文件的第3行匹配
< hello world
5d4                第一个文件的第5行删除后能和第二个文件的第4行匹配
< 333
6a6,7              第一个文件的第6行增加(a=add)内容后能和第二个文件的第6到7行匹配
> 333              需要增加的内容在第二个文件里是333和world
> world
```

2) 上下文格式显示

```
[root@MissHou ~]# diff -c file1 file2
```

前两行主要列出需要比较的文件名和文件的时间戳；文件名前面的符号***表示file1，---表示file2

```
*** file1      2019-04-16 16:26:05.748650262 +0800
--- file2      2019-04-16 16:26:30.470646030 +0800
***** 我是分隔符
*** 1,6 ****   以***开头表示file1文件，1,6表示1到6行
! aaaa         !表示该行需要修改才与第二个文件匹配
  111
- hello world  -表示需要删除该行才与第二个文件匹配
  222
- 333          -表示需要删除该行才与第二个文件匹配
  bbb
--- 1,7 ----   以---开头表示file2文件，1,7表示1到7行
! aaa          表示第一个文件需要修改才与第二个文件匹配
! hello        表示第一个文件需要修改才与第二个文件匹配
  111
  222
  bbb
+ 333          表示第一个文件需要加上该行才与第二个文件匹配
+ world        表示第一个文件需要加上该行才与第二个文件匹配
```

3) 合并格式显示

```
[root@MissHou ~]# diff -u file1 file2
```

前两行主要列出需要比较的文件名和文件的时间戳；文件名前面的符号---表示file1，+++表示file2

```
--- file1      2019-04-16 16:26:05.748650262 +0800
+++ file2      2019-04-16 16:26:30.470646030 +0800
@@ -1,6 +1,7 @@
-aaaa
+aaa
+hello
  111
-hello world
  222
-333
  bbb
+333
+world
```

- 比较两个==目录不同==

默认情况下也会比较两个目录里相同文件的内容

```
[root@MissHou tmp]# diff dir1 dir2
```

```
diff dir1/file1 dir2/file1
```

```
0a1
```

```
> hello
```

```
Only in dir1: file3
```

```
Only in dir2: test1
```

如果只需要比较两个目录里文件的不同，不需要进一步比较文件内容，需要加-q选项

```
[root@MissHou tmp]# diff -q dir1 dir2
```

```
Files dir1/file1 and dir2/file1 differ
```

```
Only in dir1: file3
```

```
Only in dir2: test1
```

其他小技巧:

有时候我们需要以一个文件为标准，去修改其他文件，并且修改的地方较多时，我们可以通过打补丁的方式完成。

1) 先找出文件不同，然后输出到一个文件

```
[root@MissHou ~]# diff -uN file1 file2 > file.patch
```

-u: 上下文模式

-N: 将不存在的文件当作空文件

2) 将不同内容打补丁到文件

```
[root@MissHou ~]# patch file1 file.patch
```

patching file file1

3) 测试验证

```
[root@MissHou ~]# diff file1 file2
```

```
[root@MissHou ~]#
```

7. paste工具

paste工具用于合并文件行

常用选项:

-d: 自定义间隔符，默认是tab

-s: 串行处理，非并行

##8. tr工具

tr用于字符转换，替换和删除；主要用于==删除文件中控制字符==或进行==字符转换==

语法和选项

语法:

用法1: 命令的执行结果交给tr处理，其中string1用于查询，string2用于转换处理

```
# commands|tr 'string1' 'string2'
```

用法2: tr处理的内容来自文件，记住要使用"<"标准输入

```
# tr 'string1' 'string2' < filename
```

用法3: 匹配string1进行相应操作，如删除操作

```
# tr options 'string1' < filename
```

常用选项:

-d 删除字符串1中所有输入字符。

-s 删除所有重复出现字符序列，只保留第一个；即将重复出现字符串压缩为一个字符串

常匹配字符串:

字符串	含义	备注
==a-z==或[:lower:]	匹配所有小写字母	[a-zA-Z0-9]
==A-Z==或[:upper:]	匹配所有大写字母	
==0-9==或[:digit:]	匹配所有数字	
[:alnum:]	匹配所有字母和数字	
[:alpha:]	匹配所有字母	
[:blank:]	所有水平空白	
[:punct:]	匹配所有标点符号	
[:space:]	所有水平或垂直的空格	

| [:cntrl:] | 所有控制字符 | \f Ctrl-L 走行换页
\n Ctrl-J 换行
\r Ctrl-M 回车
\t Ctrl-I tab键 |

举例说明:

```
[root@MissHou shell01]# cat 3.txt 自己创建该文件用于测试
ROOT:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
boss02:x:516:511:./home/boss02:/bin/bash
vip:x:517:517:./home/vip:/bin/bash
stu1:x:518:518:./home/stu1:/bin/bash
mailnull:x:47:47:./var/spool/mqueue:/sbin/nologin
smmsp:x:51:51:./var/spool/mqueue:/sbin/nologin
aaaaaaaaaaaaaaaaaaaaa
bbbbbb11111122222222222233333333cccccccc
hello world 888
666
777
999

# tr -d '[:/]' < 3.txt          删除文件中的:和/
# cat 3.txt |tr -d '[:/]'      删除文件中的:和/
# tr '[0-9]' '@' < 3.txt       将文件中的数字替换为@符号
# tr '[a-z]' '[A-Z]' < 3.txt   将文件中的小写字母替换成大写字母
# tr -s '[a-z]' < 3.txt        匹配小写字母并将重复的压缩为一个
# tr -s '[a-z0-9]' < 3.txt     匹配小写字母和数字并将重复的压缩为一个
# tr -d '[:digit:]' < 3.txt    删除文件中的数字
# tr -d '[:blank:]' < 3.txt    删除水平空白
# tr -d '[:space:]' < 3.txt    删除所有水平和垂直空白
```


小试牛刀

1. 使用小工具分别截取当前主机IP；截取NETMASK；截取广播地址；截取MAC地址

```
# ifconfig eth0|grep 'Bcast'|tr -d '[a-zA-Z ]'|cut -d: -f2,3,4
10.1.1.1:10.1.1.255:255.255.255.0
# ifconfig eth0|grep 'Bcast'|tr -d '[a-zA-Z ]'|cut -d: -f2,3,4|tr ':' '\n'
10.1.1.1
10.1.1.255
255.255.255.0
# ifconfig eth0|grep 'HWaddr'|cut -d: -f2-|cut -d' ' -f4
00:0C:29:25:AE:54
# ifconfig eth0|grep 'HW'|tr -s ' '|cut -d' ' -f5
00:0C:29:B4:9E:4E

# ifconfig eth1|grep Bcast|cut -d: -f2|cut -d' ' -f1
# ifconfig eth1|grep Bcast|cut -d: -f2|tr -d '[ a-zA-Z]'
# ifconfig eth1|grep Bcast|tr -d '[:a-zA-Z]'|tr ' ' '@'|tr -s '@'|tr '@'
'\n'|grep -v ^$
# ifconfig eth0|grep 'Bcast'|tr -d [:alpha:]|tr '[ :]' '\n'|grep -v ^$
# ifconfig eth1|grep HWaddr|cut -d' ' -f11
# ifconfig eth0|grep HWaddr|tr -s ' '|cut -d' ' -f5
# ifconfig eth1|grep HWaddr|tr -s ' '|cut -d' ' -f5
# ifconfig eth0|grep 'Bcast'|tr -d 'a-zA-Z:'|tr ' ' '\n'|grep -v '^$'
```

2. 将系统中所有普通用户的用户名、密码和默认shell保存到一个文件中，要求用户名密码和默认shell之间用tab键分割

```
# grep 'bash$' passwd |grep -v 'root'|cut -d: -f1,2,7|tr ':' '\t' |tee abc.txt
```

二、bash的特性

##1、命令和文件自动补全

Tab只能补全==命令和文件==（RHEL6/Centos6）

##2、常见的快捷键

^c	终止前台运行的程序
^z	将前台运行的程序挂起到后台
^d	退出 等价 exit
^l	清屏
^a home	光标移到命令行的最前端
^e end	光标移到命令行的后端
^u	删除光标前所有字符
^k	删除光标后所有字符
^r	搜索历史命令

##3、==常用的通配符（重点）==

```
*: 匹配0或多个任意字符
?: 匹配任意单个字符
[list]: 匹配[list]中的任意单个字符,或者一组单个字符 [a-z]
[!list]: 匹配除list中的任意单个字符
{string1,string2,...}: 匹配string1,string2或更多字符串

# rm -f file*
# cp *.conf /dir1
# touch file{1..5}
```

##4、==bash中的引号（重点）==

- 双引号"`"` :会把引号的内容当成整体来看待,允许通过\$符号引用其他变量值
- 单引号"`'`" :会把引号的内容当成整体来看待,禁止引用其他变量值, shell中特殊符号都被视为普通字符
- 反撇号"```" :反撇号和\$()一样, 引号或括号里的命令会优先执行, 如果存在嵌套, 反撇号不能用

```
[root@MissHou dir1]# echo "$(hostname)"
server
[root@MissHou dir1]# echo '$(hostname)'
$(hostname)
[root@MissHou dir1]# echo "hello world"
hello world
[root@MissHou dir1]# echo 'hello world'
hello world

[root@MissHou dir1]# echo $(date +%F)
2018-11-22
[root@MissHou dir1]# echo `echo $(date +%F)`
2018-11-22
[root@MissHou dir1]# echo `date +%F`
2018-11-22
[root@MissHou dir1]# echo `echo `date +%F``
date +%F
[root@MissHou dir1]# echo $(echo `date +%F`)
2018-11-22
```