Linux基础命令

- 在教室物理机包括rhel7和rhel6两台实验虚拟机，上打开虚拟机命令: *rht-vmctl start rhel7*
  - 在物理机上远程登陆到虚拟机rhel7 *ssh root@rhel7-fN* ， *N* 为你的机器编号， *root*密码都是*uplooking*，系统中有一个普通用户*student*，其密码为*student*。
  - 在虚拟机中关闭虚拟机的命令为*shutdown -h now*，也可以在物理机上执行*rht-vmctl poweroff rhel7*。
- 在物理机上设置录屏视频，由于Gnome3的bug会有30秒时间限制，修正时间限制的命令如下：

```
[kiosk@fundation0 ~]$ gsettings set org.gnome.settings-daemon.plugins.media-keys max-screencast-length "uint32 0"
```

命令简介

命令的构成

命令字 选项 参数

- 命令分:内部命令、外部命令;
- 选项: - 单个字符 -- 多个字符
- 参数:对谁执行这个命令,可以有多个,选项和参数可以互换位置

命令使用的原因

Shell是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行。实际上Shell是一个命令解释器，它解释由用户输入的命令并且把它们送到内核。从这里开始学习Linux命令，本课程让你更清楚地了解和掌握它，在Linux中命令是讲究大小写的。使用命令即快速又能减少机器的性能消耗。

命令提示符

- # root 用户
- $ 一般用户
- [用户的身份 @ 主机名 当前位置]

当前位置显示的是目录名

```
[root@rhel7 ~]# hostname
www.dabao.com
[root@rhel7 ~]# groupadd tom
[root@rhel7 ~]# useradd -g tom tom
[root@rhel7 ~]# id tom
uid=501(tom) gid=501(tom) groups=501(tom)
[root@rhel7 ~]# su - tom
[tom@rhel7 ~]$ who
root tty1
2016-03-20 23:56 (:0)
root pts/0
2016-03-21 00:39 (:0.0)
```

常用的命令

ls

常用参数	含义及用法
-l	long 的缩写 详细列出当前目录下的所有文件属性 七列 文件名 <=255 个字符
-d	directory 的缩写 查看当前目录本身的信息
-h	以人性化的方式显示文件大小,录的大小并不代表目录内所有文件的大小 du -sh /etc<== 查看 etc 目录真正的大小
-a	查看隐藏文件 以 . 开头的文件
-R	查看多层目录
-b	特殊字符将以 □分割 ls 查看有特殊字符的文件

ls实验

```
[tom@rhel7 tmp]$ ls -hl
total 68K
srwxr-xr-x. 1 root root 0 Mar 20 23:59 gedit.root.3177893063
drwx------. 2 root root 4.0K Feb 5 18:24 keyring-JL7MKY
[tom@rhel7 tmp]$ ls -ld /tmp
drwxrwxrwt. 22 root root 4096 Mar 21 00:43 /tmp
[tom@rhel7 tmp]$ ls -la /tmp
total 96
drwxrwxrwt. 22 root root 4096 Mar 21 00:43 .
dr-xr-xr-x. 29 root root 4096 Mar 20 23:55 ..
drwx------. 2 root root 4096 Mar 20 23:56 .esd-0
drwx------. 2 cong cong 4096 Jan 1 18:33 .esd-500
[tom@rhel7 tmp]$ ls -bl
-rw-rw-r--. 1 tom tom 0 Mar 21 00:48 a\ b.txt
[tom@rhel7 tmp]$ ls -l
-rw-rw-r--. 1 tom tom 0 Mar 21 00:48 a b.txt
[tom@rhel7 tmp]$ mkdir -p a/b/c
[tom@rhel7 tmp]$ ls -lR a/
a/:
total 4
drwxrwxr-x. 3 tom tom 4096 Mar 21 00:50 b
a/b:
total 4
drwxrwxr-x. 2 tom tom 4096 Mar 21 00:50 c
a/b/c:
total 0
```

cd

cd	**change directory** 切换工作目录
绝对路径	以根为起始的路径
相对路径	~当前用户的家目录 ;. 当前目录 ;.. 上一层用户 ;- 回到上一次所在位置

cd实验

```
[tom@rhel7 tmp]$ cd /etc/
[tom@rhel7 etc]$ cd ~
[tom@rhel7 ~]$ cd .
[tom@rhel7  ~]$ cd ..
[tom@rhel7  home]$ cd -
/home/tom
```

pwd

pwd:print working directory 显示当前所在位置的绝对路径

pwd实验

```
[tom@rhel7 tmp]$ cd /etc/nginx
[tom@rhel7 nginx]$ cd conf.d
[tom@rhel7 conf.d]$ pwd
/etc/nginx/conf.d
```

符号

通配符 *

匹配任意多个字符

例如：a* 包括aa*、ab*、ac* 等等

通配符 ?

匹配任意单个字符
例如： a? 可以是ab、ac、ad、a1、a9、a#等等

*? 实验

```
rm -f *1
rm -f 1*
rm -f 1*1
rm -f test?
想删除 test 后面有一个字符的文件
```

| 管道

output | input

对某些命令执行的结果去作操作,会用到管道

| 管道实验

详细列出 /tmp 目录下的文件,并截取以空格为分割的第三列

```
[tom@rhel7 ~]$ ls -l /tmp|cut -d" " -f3
tom
tom
root
root
root
详细列出 /tmp 目录下的文件,截取含有关键字 tom 的行,再截取以空格为分割的第一列内容
[tom@rhel7 ~]$ ls -l /tmp|grep tom|cut -d" " -f1
drwxrwxr-x.
-rw-rw-r--.
```

针对文件的的基本操作

touch

touch [filename]

创建文件,参数可以跟多个

如果要创建 50 个有规律的文件,例如 text1-text50

利用参数扩展

```
touch test{1..50}
touch test{a..e}
touch test{a..e}_{1..3}---> 会创建 a_1 a_2 a_3...
```

上帝之手,本来是用来修改文件时间戳的。文件的三个时间 ctime\mtime\atime

拓展内容：可以通过“stat”命令查看文件的三个时间

touch " " 可以放一些特殊字符

touch实验

```
[tom@rhel7 ~]$ touch test{a..c}_{1..4}
[tom@rhel7 ~]$ ls
testa_1 testa_4 testb_3 testc_2
testa_2 testb_1 testb_4 testc_3
testa_3 testb_2 testc_1 testc_4

--full-time可以查看mtime的完整时间

[tom@rhel7 ~]$ ls -l --full-time
total 0
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testa_1
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testa_2
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testa_3
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testa_4
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testb_1
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testb_2
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testb_3
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.853039590 +0800 testb_4
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.854039544 +0800 testc_1
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.854039544 +0800 testc_2
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.854039544 +0800 testc_3
-rw-rw-r--. 1 tom tom 0 2016-03-21 01:31:22.854039544 +0800 testc_4

[tom@rhel7 ~]$ touch "ab cd"

[tom@rhel7 ~]$ ls -b
ab\ \ \ cd testa_3 testb_2 testc_1 testc_4
testa_1 testa_4 testb_3 testc_2
testa_2 testb_1 testb_4 testc_3
```

touch拓展实验

```
[booboo@rhel7 ~]$ touch booboo
[booboo@rhel7 ~]$ ll
total 0
-rw-rw-r--. 1 booboo booboo 0 Jun 15 23:28 booboo
[booboo@rhel7 ~]$ stat booboo
  File: ‘booboo’
  Size: 0           Blocks: 0          IO Block: 4096   regular empty file
Device: fd01h/64769d    Inode: 143         Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1001/  booboo)   Gid: ( 1001/  booboo)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2016-06-15 23:28:55.041578819 -0400     #atime 文件最近一次被访问的时间
Modify: 2016-06-15 23:28:55.041578819 -0400    #mtime 文件内容最近一次修改的时间
Change: 2016-06-15 23:28:55.041578819 -0400    #ctime 文件属性最近一次修改的时间
 Birth: -
```

使用cat去访问booboo文件，可以发现atime被修改了

```
[booboo@rhel7 ~]$ cat booboo
[booboo@rhel7 ~]$ stat booboo
  File: ‘booboo’
  Size: 0           Blocks: 0          IO Block: 4096   regular empty file
Device: fd01h/64769d    Inode: 143         Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1001/  booboo)   Gid: ( 1001/  booboo)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2016-06-15 23:32:35.898724748 -0400
Modify: 2016-06-15 23:28:55.041578819 -0400
Change: 2016-06-15 23:28:55.041578819 -0400
 Birth: -
```

通过chmod修改文件权限后，会看到ctime时间改变，通过ll命令看到的时间为mtime

```
[booboo@rhel7 ~]$ chmod 777 booboo
[booboo@rhel7 ~]$ ll
total 0
-rwxrwxrwx. 1 booboo booboo 0 Jun 15 23:28 booboo
[booboo@rhel7 ~]$ stat booboo
  File: ‘booboo’
  Size: 0           Blocks: 0          IO Block: 4096   regular empty file
Device: fd01h/64769d    Inode: 143         Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 1001/  booboo)   Gid: ( 1001/  booboo)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2016-06-15 23:32:35.898724748 -0400
Modify: 2016-06-15 23:28:55.041578819 -0400
Change: 2016-06-15 23:33:49.195445761 -0400
 Birth: -
```

通过echo命令向booboo文件追加一些内容，会看到mtime时间变了，并且ctime也变了，思考为什么？

```
[booboo@rhel7 ~]$ echo hi >> booboo
[booboo@rhel7 ~]$ ll
total 4
-rwxrwxrwx. 1 booboo booboo 3 Jun 15 23:34 booboo
[booboo@rhel7 ~]$ stat booboo
  File: ‘booboo’
  Size: 3           Blocks: 8          IO Block: 4096   regular file
Device: fd01h/64769d    Inode: 143         Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 1001/  booboo)   Gid: ( 1001/  booboo)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2016-06-15 23:32:35.898724748 -0400
Modify: 2016-06-15 23:34:53.251332183 -0400
Change: 2016-06-15 23:34:53.251332183 -0400
 Birth: -
```

rm

rm	**[filename] remove** 删除文件,对 **root** 用户有提示,普通用户没有提示
-f	force 强制删除, root 无提示
-i	普通用户有提示的删除
-r	递归删除,慎重使用 -rf

rm实验

普通用户 tom

```
[tom@rhel7 ~]$ rm testa_1
[tom@rhel7 ~]$ rm -i testa_2
rm: remove regular empty file `testa_2'? Y
```

root 用户

```
[root@rhel7 ~]# rm a.test
rm: remove regular empty file `a.test'? Y
[root@rhel7 ~]# rm -f b.test
```

普通用户 tom

```
[tom@rhel7 ~]$ mkdir -p a/b/c
[tom@rhel7 ~]$ rm a/
rm: cannot remove `a/': Is a directory
[tom@rhel7 ~]$ rm -r a/
```

mkdir

mkdir:make directory 创建目录

`mkdir -p /test/test1`

第归创建目录

`mkdir {a..e}`

创建 a-e 的目录

`touch {a..e}/file{1..4}`

在 a-e 的目录下新建 file1-file4 文件

mkdir实验

```
[tom@rhel7 ~]$ mkdir -p test/test1
[tom@rhel7 ~]$ ls
ab cd testa_3 testb_2 testc_1 testc_4
p        testa_4 testb_3 testc_2
test    testb_1 testb_4 testc_3
[tom@rhel7 ~]$ mkdir {a..f}
[tom@rhel7 ~]$ ls
a d test testb_2 testc_2
ab cd e testa_3 testb_3 testc_3
b f testa_4 testb_4 testc_4 c p testb_1 testc_1
[tom@rhel7 ~]$ touch {a..f}/file{1..4}
[tom@rhel7 ~]$ ls
a d test testb_2 testc_2
ab cd e testa_3 testb_3 testc_3
b f testa_4 testb_4 testc_4 c p testb_1 testc_1
[tom@rhel7 ~]$ ls -rR
.:
testc_4 testb_4 testa_4 f b
testc_3 testb_3 testa_3 e ab cd
testc_2 testb_2 test d a
testc_1 testb_1 p c
./test:
test1
./test/test1:
./p:
./f:
file4 file3 file2 file1
./e:
file4 file3 file2 file1
./d:
file4 file3 file2 file1
./c:
file4 file3 file2 file1
./b:
file4 file3 file2 file1
./a:
file4 file3 file2 file1
```

rmdir

rmdir:remove directory 删除目录

只能删除空目录,出于安全性的考虑

`rm -rf [d_name]`

可以删除空目录

rmdir 实验

```
[booboo@rhel7 ~]$ mkdir aa
[booboo@rhel7 ~]$ mkdir -p cc/bb
[booboo@rhel7 ~]$ rmdir aa
[booboo@rhel7 ~]$ ll
total 4
-rwxrwxrwx. 1 booboo booboo  3 Jun 15 23:34 booboo
drwxrwxr-x. 3 booboo booboo 15 Jun 16 00:08 cc
[booboo@rhel7 ~]$ rmdir cc
rmdir: failed to remove ‘cc’: Directory not empty
[booboo@rhel7 ~]$ rm -rf cc
[booboo@rhel7 ~]$ ll
total 4
-rwxrwxrwx. 1 booboo booboo 3 Jun 15 23:34 booboo
```

cp

cp:copy 复制文件

cp 源文件 目的地(目录)

-p 保留文件原属性

-r 复制目录

cp实验

```
使用 root 用户,进入 tom 的家目录

[root@rhel7 ~]# cd /home/tom

保留原文件属性复制 testa_3 文件到 /tmp 目录下

[root@rhel7 tom]# cp -p testa_3 /tmp
[root@rhel7 tom]# ll /tmp/testa_3
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 /tmp/testa_3
[root@rhel7 tom]# cp testa_4 /tmp
[root@rhel7 tom]# ll /tmp/testa_4
-rw-r--r--. 1 root root 0 Mar 21 01:55 /tmp/testa_4

拷贝目录 a 到 /tmp 下,必须加上 -r ,因为 a 目录下还有 b/c 目录

[root@rhel7 tom]# cp a /tmp
cp: omitting directory `a'
[root@rhel7 tom]# cp -r a /tmp
```

mv

mv:move 移动 移动和重命名

mv 源文件 目的地(目录)

mv实验

```
[tom@rhel7 ~]$ mv testa_3 test
[tom@rhel7 ~]$ ls
a
c f testa_4 testb_3 testc_2
ab cd d p testb_1 testb_4 testc_3
b
e test testb_2 testc_1 testc_4
[tom@rhel7 ~]$ mv test??? f/
[tom@rhel7 ~]$ ll -R f
f:
total 0
-rw-rw-r--. 1 tom tom 0 Mar 21 01:47 file1
-rw-rw-r--. 1 tom tom 0 Mar 21 01:47 file2
-rw-rw-r--. 1 tom tom 0 Mar 21 01:47 file3
-rw-rw-r--. 1 tom tom 0 Mar 21 01:47 file4
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testa_4
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testb_1
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testb_2
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testb_3
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testb_4
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testc_1
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testc_2
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testc_3
-rw-rw-r--. 1 tom tom 0 Mar 21 01:31 testc_4
```

针对文件内容的基本操作

文件的查看

文件	命令	解释
小文件	cat	以正序查看 调用内存比较多 -n 显示行号
	tac	以倒序查看 调用内存比较多
大文件	head	查看文件首部，默认10行 ， -n 指定行号
	tail	查看文件尾部，默认10行，-n指定行号
	more	按空格 space 下一页 b 向上翻页 enter 下一行
	less	比 more 多了一个搜索功能 /[需搜索的子段]N 向上查找 n 向下查 q 退出

文件查看实验

```
[tom@rhel7 ~]$ cat -n passwd
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
6 sync:x:5:0:sync:/sbin:/bin/sync
7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8 halt:x:7:0:halt:/sbin:/sbin/halt
9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
10 cong:x:500:500:cong:/home/cong:/bin/bash
11 nginx:x:496:491:Nginx web server:/var/lib/nginx:/sbin/nologin
12 tom:x:501:501::/home/tom:/bin/bash
[tom@rhel7 ~]$ tac passwd
tom:x:501:501::/home/tom:/bin/bash
nginx:x:496:491:Nginx web server:/var/lib/nginx:/sbin/nologin
cong:x:500:500:cong:/home/cong:/bin/bash
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
halt:x:7:0:halt:/sbin:/sbin/halt
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
sync:x:5:0:sync:/sbin:/bin/sync
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
```

文件的修改

软件	解释
LibreOffice	.odt 结尾 类似于 windows office
gedit	类似于 windows 记事本
vim	插入模式 后面会专门讲到 vim 编辑器的使用 退出模式 命令模式
echo	本身代表回显 echo xxx > file 将 xxx 写入 file 文件,并覆盖原有内容 echo xxx >> file 在 file 文件追加

echo实验

```
[booboo@rhel7 ~]$ echo hi > file1
[booboo@rhel7 ~]$ cat file1
hi
[booboo@rhel7 ~]$ echo booboo > file1
[booboo@rhel7 ~]$ cat file1
booboo
[booboo@rhel7 ~]$ echo hihihihi >> file1
[booboo@rhel7 ~]$ cat file1
booboo
hihihihi
```

文件的过滤

grep

截取行

```
grep [OPTIONS] PATTERN [FILE...]
过滤带有 [ 字符串 ] 的行
grep [ 字符串 ] [ 文件 ]
过滤以 [ 字符串 ] 为开始的行
grep [^ 字符串 ] [ 文件 ]
过滤以 [ 字符串 ] 为结尾的行
grep [ 字符串 $] [ 文件 ]
过滤反选
grep -v [ 字符串 ] [ 文件 ]

eg.
过滤以 root 为开始的行
 grep ^root /etc/passwd
过滤以 bash 为结尾的行
grep bash$ /etc/passwd
```

cut

截取列

```
cut -d" 分割符 "( 以什么为分隔符 ) -fn( 第几列 ) [ 文件 ]
eg. cut -d":" -f2 /etc/resolv.conf
```

wc

统计

```
wc 行数 单词数 字符数 文件名
-l 只显示行数
-w, --words 显示单词数
-c, -m,--bytes 显示字节
eg.
[root@stu15 ~]# wc /etc/resolv.conf 、
4 11 98 /etc/resolv.conf
```

sort

排序

```
默认按照首字母 ACII 码
-n 按照数字大小排序
-u 剔除重复的行
-r 降序排列
-k 指定某一列
-t 分隔符
eg.sort -n -k 2 -t : file1
将 file1 以第二列的数字排序,列以:分割
```

uniq

剔除重复行

```
uniq [file_name]
```

文件过滤实验

grep

```
[booboo@rhel7 ~]$ grep root passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[booboo@rhel7 ~]$ grep ^booboo passwd
booboo:x:1001:1001::/home/booboo:/bin/bash
[booboo@rhel7 ~]$ grep ^root passwd
root:x:0:0:root:/root:/bin/bash
[booboo@rhel7 ~]$ grep bash$ passwd
root:x:0:0:root:/root:/bin/bash
student:x:1000:1000:student:/home/student:/bin/bash
booboo:x:1001:1001::/home/booboo:/bin/bash
[booboo@rhel7 ~]$ grep -v nologin passwd
root:x:0:0:root:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
student:x:1000:1000:student:/home/student:/bin/bash
booboo:x:1001:1001::/home/booboo:/bin/bash
```

cut

```
[booboo@rhel7 ~]$ cut -d":" -f1 passwd
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
```

wc

```
[booboo@rhel7 ~]$ wc passwd
  42   72 2121 passwd
[booboo@rhel7 ~]$ wc -l passwd
42 passwd
[booboo@rhel7 ~]$ wc -w passwd
72 passwd
[booboo@rhel7 ~]$ wc -c passwd
2121 passwd
```

sort

```
[booboo@rhel7 ~]$ vi num
[booboo@rhel7 ~]$ cat num
booboo 20 100
tom 21 100
```

```
tom 21 100
kevin 19 200
booboo 20 100
mark 20 200
[booboo@rhel7 ~]$ sort num
booboo 20 100
booboo 20 100
kevin 19 200
mark 20 200
tom 21 100
tom 21 100
[booboo@rhel7 ~]$ sort -r num
tom 21 100
tom 21 100
mark 20 200
kevin 19 200
booboo 20 100
booboo 20 100
[booboo@rhel7 ~]$ sort -u num
booboo 20 100
kevin 19 200
mark 20 200
tom 21 100
[booboo@rhel7 ~]$ sort -n -k 2 -t " " num |sort -u
booboo 20 100
kevin 19 200
mark 20 200
tom 21 100

uniq

[booboo@rhel7 ~]$ uniq num
booboo 20 100
tom 21 100
kevin 19 200
booboo 20 100
mark 20 200
```

帮助命令

命令	解释
type [命令]	判断是内部命令 or 外部命令
--help	外部命令
help	只针对系统内部命令
man []	内容清晰、详细,在线文档,支持搜索(/name) `man [ 章节 ] [name]`
info []	太详细
/usr/share/doc	存放帮助文档,在与软件同名的目录下有所有软件的使用文档

man和**—help**以及**help**的区别

man命令

系统中会有单独的man文件，就是说，如果系统没有安装对应man文件，哪怕命令完全正常，man都没结果（同样，只要安装了man文件，哪怕没命令，也可以得到一大堆东西）。

--help参数
将会显示可执行程序自带的信息，这些信息是嵌入到程序本身的，所以--help信息较简短。

help命令
是选项帮助命令，顾名思义 你可以把单独某个命令的某个选项列出来，方便快捷很多，省去了man当中查找的繁琐，但是help只支持shell的内部命令。内部命令即存储在shell内部可以直接调用的一些简单命令，比如说echo，cd，pwd等。

type

type命令用来显示指定命令的类型，判断给出的指令是内部指令还是外部指令。

```
语法
type(选项)(参数)
选项
-t：输出“file”、“alias”或者“builtin”，分别表示给定的指令为“外部指令”、“命令别名”或者“内部指令”；
-p：如果给出的指令为外部指令，则显示其绝对路径；
-a：在环境变量“PATH”指定的路径中，显示给定指令的信息，包括命令别名。 参数 指令：要显示类型的指令。
参数
关键字：指定要搜索帮助的关键字
命令类型
alias：别名
keyword：关键字，Shell保留字
function：函数，Shell函数
builtin：内建命令，Shell内建命令
file：文件，磁盘文件，外部命令
unfound：没有找到

尝试几个命令man,ls,touch,echo,cat

[booboo@rhel7 ~]$ type man
man is /bin/man
[booboo@rhel7 ~]$ type ls
ls is aliased to `ls -color=auto'
[booboo@rhel7 ~]$ which ls
alias ls='ls --color=auto'
    /bin/ls
[booboo@rhel7 ~]$ type touch
touch is hashed (/bin/touch)
[booboo@rhel7 ~]$ type echo
echo is a shell builtin
[booboo@rhel7 ~]$ type cat
cat is hashed (/bin/cat)
```

内置命令一般显示为“is a shell builtin ”

除了内置命令外，其他命令都是通过某个安装包安装生成的可执行文件

接下来我们一起看看这些命令都是什么软件生成的

```
[booboo@rhel7 ~]$ rpm -qf /bin/man
man-db-2.6.3-9.el7.x86_64
[booboo@rhel7 ~]$ rpm -qf /bin/ls
coreutils-8.22-11.el7.x86_64
[booboo@rhel7 ~]$ rpm -qf /bin/touch
coreutils-8.22-11.el7.x86_64
[booboo@rhel7 ~]$ rpm -qf /bin/cat
coreutils-8.22-11.el7.x86_64
[booboo@rhel7 ~]$ type if
if is a shell keyword
```

--help 参数

--help参数是大所数命令自带的选项，用于查看使用帮助。

```
[booboo@rhel7 ~]$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      scale sizes by SIZE before printing them; e.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'never', 'auto',
                               or 'always' (the default); more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
```

help

help只支持shell的内部命令。内部命令即存储在shell内部可以直接调用的一些简单命令,例如cd,echo,help等。

help(选项)(参数)
-s：输出短格式的帮助信息。仅包括命令格式。

```
[booboo@rhel7 ~]$ type cd
cd is a shell builtin
[booboo@rhel7 ~]$ help cd
cd: cd [-L|[-P [-e]]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is the value of the
    HOME shell variable.

    The variable CDPATH defines the search path for the directory containing
    DIR.  Alternative directory names in CDPATH are separated by a colon (:).
    A null directory name is the same as the current directory.  If DIR begins
    with a slash (/), then CDPATH is not used.

    If the directory is not found, and the shell option `cdable_vars' is set,
    the word is assumed to be  a variable name.  If that variable has a value,
    its value is used for DIR.

    Options:
        -L  force symbolic links to be followed
        -P  use the physical directory structure without following symbolic
        links
        -e  if the -P option is supplied, and the current working directory
        cannot be determined successfully, exit with a non-zero status

    The default is to follow symbolic links, as if `-L' were specified.

    Exit Status:
    Returns 0 if the directory is changed, and if $PWD is set successfully when
    -P is used; non-zero otherwise.
[booboo@rhel7 ~]$ type touch
touch is hashed (/bin/touch)
[booboo@rhel7 ~]$ type echo
echo is a shell builtin
[booboo@rhel7 ~]$ help echo
echo: echo [-neE] [arg ...]
```

man

man命令是Linux下的帮助指令，通过man指令可以查看Linux中的指令帮助、配置文件帮助和编程帮助等信息。

```
语法
man(选项)(参数)
选项
-a：在所有的man帮助手册中搜索；
-f：等价于whatis指令，显示给定关键字的简短描述信息；
-P：指定内容时使用分页程序；
-M：指定man手册搜索的路径。
参数
数字：指定从哪本man手册中搜索帮助
关键字：指定要搜索帮助的关键字
```

例如输入man ls，它会在左上角显示“ECHO(1)”“ECHO”代表手册名称；“(1)”代表 表示该手册位于第一节章1）”表示该手册位于第一节章，同样，我们输man ifconfig它会在最左上角显示“IFCONFIG（8）”。也可以这样输入命令：“man [章节号] 手册名称”。 man是按照手册的章节号的顺序进行搜索的，比如： man sleep 只会显示sleep命令的手册,如果想查看库函数sleep，就要输入: man 3 sleep

```
[booboo@rhel7 ~]$ man echo|cat
ECHO(1)                          User Commands                          ECHO(1)
#此处“ECHO”代表手册名称；“（1）”代表 表示该手册位于第一节章



NAME
       echo - display a line of text

SYNOPSIS
       echo [SHORT-OPTION]... [STRING]...
       echo LONG-OPTION

DESCRIPTION
       Echo the STRING(s) to standard output.

       -n     do not output the trailing newline

       -e     enable interpretation of backslash escapes

       -E     disable interpretation of backslash escapes (default)

       --help display this help and exit

       --version
              output version information and exit

       If -e is in effect, the following sequences are recognized:

       \\     backslash

       \a     alert (BEL)

       \b     backspace

       \c     produce no further output

       \e     escape

       \f     form feed

       \n     new line

       \r     carriage return

       \t     horizontal tab

       \v     vertical tab

       \0NNN  byte with octal value NNN (1 to 3 digits)

       \xHH   byte with hexadecimal value HH (1 to 2 digits)
```

```
       NOTE:  your  shell  may  have  its  own version of echo, which usually
       supersedes the version described here.  Please refer to  your  shell's
       documentation for details about the options it supports.

       GNU  coreutils  online  help: <http://www.gnu.org/software/coreutils/>
       Report echo translation bugs to <http://translationproject.org/team/>
AUTHOR
       Written by Brian Fox and Chet Ramey.
COPYRIGHT
       Copyright © 2013 Free Software Foundation, Inc.  License  GPLv3+:  GNU
       GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
       This  is  free  software:  you are free to change and redistribute it.
       There is NO WARRANTY, to the extent permitted by law.
SEE ALSO
       The full documentation for echo is maintained as a Texinfo manual.  If
       the  info  and  echo programs are properly installed at your site, the
       command
              info coreutils 'echo invocation'
      should give you access to the complete manual.

GNU coreutils 8.22              January 2014                       ECHO(1)
```

info（作为拓展内容）

info命令是Linux下info格式的帮助指令。 就内容来说，info页面比man page编写得要更好、更容易理解，也更友好，但man page使用起来确实要更容易得多。一个man page只有一页，而info页面几乎总是将它们的内容组织成多个区段（称为节点），每个区段也可能包含子区段（称为子节点）。理解这个命令的窍门就是不仅要学习如何在单独的Info页面中浏览导航，还要学习如何在节点和子节点之间切换。可能刚开始会一时很难在info页面的节点之间移动和找到你要的东西，真是具有讽刺意味：原本以为对于新手来说，某个东西比man命令会更好些，但实际上学习和使用起来更困难。

```
语法
info(选项)(参数)
选项
-d：添加包含info格式帮助文档的目录；
 -f：指定要读取的info格式的帮助文档；
-n：指定首先访问的info帮助文件的节点；
 -o：输出被选择的节点内容到指定文件。
参数
 帮助主题：指定需要获得帮助的主题，可以是指令、函数以及配置文件。
常用快捷键
?键：它就会显示info的常用快捷键。
N键：显示（相对于本节点的）下一节点的文档内容。
P键：显示（相对于本节点的）前一节点的文档内容。
U键：进入当前命令所在的主题。
M键：敲M键后输入命令的名称就可以查看该命令的帮助文档了。
G键：敲G键后输入主题名称，进入该主题。
L键：回到上一个访问的页面。 SPACE键：向前滚动一页。
BACKUP或DEL键：向后滚动一页。
Q：退出info。
命令
？ 显示帮助窗口
在帮助窗口中：
Ctrl-x 0 关闭帮助窗口
Ctrl-x Ctrl-c 关闭整个 Info
q 退出 info
n 打开与本 Node 关联的下一个 Node
p 打开与本 Node 关联的前一个 Node
u 打开与本 Node 关联的上一个 Node
l 回到上一次访问的 Node
m或g 选择一个菜单项（Node 的名字）
           输入指定菜单的名字后按回车，打开指定菜单项关联的 Node
空格键 下一页（PageDown 也可以，下一页从当前页的最后两行开始算起）
           下一个 Node （若当前页在 Node 文档的末尾）
Del 键 上一页（PageUp 也可以，上一页从当前页的开始两行开始算起）
       上一个 Node （若当前页 Node 文档的开始）
b 或 t 或 Home 文档的开始（b 是 begining 的意思）
e 或 End 文档的末尾（b 是 ending 的意思）
Ctrl-l 刷新当前页，若当前文档显示情况有问题时
Ctrl-g 取消所键入的指令
```

关于时间的命令

date

date命令是显示或设置系统时间与日期。 很多shell脚本里面需要打印不同格式的时间或日期，以及要根据时间和日期执行操作。延时通常用于脚本执行过程中提供一段等待的时间。日期可以以多种格式去打印，也可以使用命令设置固定的格式。在类UNIX系统中，日期被存储为一个整数，其大小为自世界标准时间（UTC）1970年1月1日0时0分0秒起流逝的秒数。

语法
date(选项)(参数)
选项
-d<字符串>：显示字符串所指的日期与时间。字符串前后必须加上双引号；
-s<字符串>：根据字符串来设置日期与时间。字符串前后必须加上双引号；
 -u：显示GMT；
--help：在线帮助；
--version：显示版本信息。
参数
<+时间日期格式>：指定显示时使用的日期时间格式。

日期格式字符串列表

格式	说明
%H	小时，24小时制（00~23）
%I	小时，12小时制（01~12）
%k	小时，24小时制（0~23）
%l	小时，12小时制（1~12）
%M	分钟（00~59）
%p	显示出AM或PM
%r	显示时间，12小时制（hh:mm:ss %p）
%s	从1970年1月1日00:00:00到目前经历的秒数
%S	显示秒（00~59）
%T	显示时间，24小时制（hh:mm:ss）
%X	显示时间的格式（%H:%M:%S）
%Z	显示时区，日期域（CST）
%a	星期的简称（Sun~Sat）
%A	星期的全称（Sunday~Saturday）
%h,%b	月的简称（Jan~Dec）
%B	月的全称（January~December）
%c	日期和时间（Tue Nov 20 14:12:58 2012）
%d	一个月的第几天（01~31）
%x,%D	日期（mm/dd/yy）
%j	一年的第几天（001~366）
%m	月份（01~12）
%w	一个星期的第几天（0代表星期天）
%W	一年的第几个星期（00~53，星期一为第一天）
%y	年的最后两个数字（1999则是99）
%Y	年1999

date 实验

```
格式化输出
[booboo@rhel7 ~]$ date +"%y/%m/%d"
16/06/16
[booboo@rhel7 ~]$ date +"%Y/%m/%d"
2016/06/16
输出昨天或后天的日期
[booboo@rhel7 ~]$ date -d "1 day ago" +"%Y-%m-%d"
2016-06-15
[booboo@rhel7 ~]$ date -d "-1 day" +%Y%m%d
20160615
[booboo@rhel7 ~]$ date -d "+1 day" +%Y%m%d
20160617
输出50秒后的时间
[booboo@rhel7 ~]$ date -d "50 second" +"%Y-%m-%d %H:%M:%S"
2016-06-16 02:52:41
传说中的1234567890秒
[booboo@rhel7 ~]$ date -d "1970-01-01 1234567890 seconds" +"%Y-%m-%d %H:%M:%S"
2009-02-13 23:31:30
2009年2月13日星期五，协调世界时（UTC）晚上11:31:30，UNIX时间将抵达1234567890秒。
UNIX时间是UNIX或类UNIX系统使用的时间表示方式：从协调世界时1970年1月1日0时0分0秒起至现在的总秒数，不包括闰秒。由于大部分UNIX的系统都是32位，因此到2038年时间计数就可能溢出，解决方法是更换为64位模式。Linux内核开发者Alan Cox表示，Linux现在都运行64位时间模式，它可以记录到2900亿年后，因此即使太阳燃料用尽也不会出问题。
普通格式转化
[booboo@rhel7 ~]$ date -d "2016-06-16" +"%Y/%m/%d %H:%M:%S"
2016/06/16 00:00:00
设定时间需要root用户权限，此处用booboo用户模拟，没有真的修改时间
[booboo@rhel7 ~]$ date
Thu Jun 16 03:01:19 EDT 2016
[booboo@rhel7 ~]$ date -s "02:03:08 2018-05-09"
date: cannot set date: Operation not permitted
Wed May  9 02:03:08 EDT 2018
[booboo@rhel7 ~]$ date -s "02:03:08 20180509"
date: cannot set date: Operation not permitted
Wed May  9 02:03:08 EDT 2018
[booboo@rhel7 ~]$ date -s "20180509 15:02:02"
date: cannot set date: Operation not permitted
Wed May  9 15:02:02 EDT 2018
[booboo@rhel7 ~]$ date -s "2018-05-09 15:02:02"
date: cannot set date: Operation not permitted
Wed May  9 15:02:02 EDT 2018
[booboo@rhel7 ~]$ date -s "2019/05/09 15:02:02"
date: cannot set date: Operation not permitted
Thu May  9 15:02:02 EDT 2019
```

date 拓展实验

1) 有时需要检查一组命令花费的时间 比如

```
ping执行的时间
[root@rhel7 ~]# vi a.sh
[root@rhel7 ~]# cat a.sh
#!/bin/bash
start=$(date +%s)
ping -c 10 172.25.0.11 &> /dev/null
end=$(date +%s)
difference=$(( $end - $start ))
echo $difference seconds.
[root@rhel7 ~]# bash a.sh
9 seconds.
sql脚本运行的是时间
#!/bin/bash
start=$(date +%s)
$mysql< mysql.all.sql &> /dev/null
end=$(date +%s)
difference=$(( $end - $start ))
echo $difference seconds.
```

2) 创建以当前时间为文件名的目录

```
[root@rhel7 ~]# mkdir `date +%Y%m%d`
[root@rhel7 ~]# ls
20160616
```

3）备份以时间做为文件名的

```
[root@rhel7 ~]# tar -cvf `date +%Y%m%d`.tar ./`date +%Y%m%d`
./20160616/
[root@rhel7 ~]# ls
20160616         Documents                          Pictures
20160616.tar
```

4）编写shell脚本计算离自己生日还有多少天？

```
[root@rhel7 ~]# cat bb.sh
#!/bin/bash
read -p "Input your birthday(YYYYmmdd):" date1
m=`date -d "$date1" +%m`    #得到生日的月
d=`date -d "$date1" +%d`    #得到生日的日
date_now=`date +%s`             #得到当前时间的秒值
y=`date +%Y`                    #得到当前时间的年
birth=`date -d "$y$m$d" +%s`      #得到今年的生日日期的秒值
internal=$(($birth-$date_now))        #计算今日到生日日期的间隔时间
if [ "$internal" -lt "0" ]; then             #判断今天的生日是否已过
        birth=`date --date="$(($y+1))$m$d" +%s`       #得到明年的生日日期秒值
        internal=$(($birth-$date_now))                #计算今天到下一个生日的间隔时间
fi
        echo "There is :$(($internal/60/60/24)) days."       #输出结果，秒换算为天
Input your birthday(YYYYmmdd):19960506
There is :323 days.
[root@rhel7 ~]# bash bb.sh
Input your birthday(YYYYmmdd):19960902
There is :77 days.
```

hwclock

hwclock命令是一个硬件时钟访问工具，它可以显示当前时间、设置硬件时钟的时间和设置硬件时钟为系统时间，也可设置系统时间为硬件时钟的时间。 在Linux中有硬件时钟与系统时钟等两种时钟。硬件时钟是指主机板上的时钟设备，也就是通常可在BIOS画面设定的时钟。系统时钟则是指kernel中的时钟。当Linux启动时，系统时钟会去读取硬件时钟的设定，之后系统时钟即独立运作。所有Linux相关指令与函数都是读取系统时钟的设定。

查看 BIOS 时间

修改 BIOS 时间

`hwclock --systohc` 将硬件时间与系统时间同步,以系统为基准

`hwclock --hctosys` 将系统时间与硬件时间同步,以硬件为基准

timedatectl

比 date 多了时区的功能

rhel6修改时区

```
[booboo@rhel7 ~]$ tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent or ocean.
 1) Africa
 2) Americas
 3) Antarctica
 4) Arctic Ocean
 5) Asia
 6) Atlantic Ocean
 7) Australia
 8) Europe
 9) Indian Ocean
10) Pacific Ocean
11) none - I want to specify the time zone using the Posix TZ format.
\#? 5
Please select a country.
 1) Afghanistan       18) Israel            35) Palestine
 2) Armenia           19) Japan             36) Philippines
 3) Azerbaijan        20) Jordan            37) Qatar
 4) Bahrain           21) Kazakhstan        38) Russia
 5) Bangladesh        22) Korea (North)     39) Saudi Arabia
 6) Bhutan            23) Korea (South)     40) Singapore
 7) Brunei            24) Kuwait            41) Sri Lanka
 8) Cambodia          25) Kyrgyzstan        42) Syria
 9) China             26) Laos              43) Taiwan
10) Cyprus            27) Lebanon           44) Tajikistan
11) East Timor        28) Macau             45) Thailand
12) Georgia           29) Malaysia          46) Turkmenistan
13) Hong Kong         30) Mongolia          47) United Arab Emirates
14) India             31) Myanmar (Burma)   48) Uzbekistan
15) Indonesia         32) Nepal             49) Vietnam
16) Iran              33) Oman              50) Yemen
17) Iraq              34) Pakistan
\#? 9
Please select one of the following time zone regions.
1) east China - Beijing, Guangdong, Shanghai, etc.
2) Heilongjiang (except Mohe), Jilin
3) central China - Sichuan, Yunnan, Guangxi, Shaanxi, Guizhou, etc.
4) most of Tibet & Xinjiang
5) west Tibet & Xinjiang
\#? 1

The following information has been given:

    China
    east China - Beijing, Guangdong, Shanghai, etc.

Therefore TZ='Asia/Shanghai' will be used.
Local time is now:  Thu Jun 16 15:08:57 CST 2016.
Universal Time is now:  Thu Jun 16 07:08:57 UTC 2016.
Is the above information OK?
1) Yes
2) No
```

```
\#? 1

You can make this change permanent for yourself by appending the line
    TZ='Asia/Shanghai'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /bin/tzselect command in shell scripts:
Asia/Shanghai
```

rhel7 版本

```
[kiosk@foundation0 Desktop]$ timedatectl
      Local time: Wed 2016-06-15 16:47:40 CST
  Universal time: Wed 2016-06-15 08:47:40 UTC
        RTC time: Wed 2016-06-15 16:47:40
       Time zone: Asia/Shanghai (CST, +0800)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: yes
      DST active: n/a
```

关键词：

```
UTC 协调世界时 Coordinated Universal Time
    又称世界统一时间，世界标准时间，国际协调时间。1972 年1 月1日，UTC（协调世界时）成为新的世界标准时间。
CST 时区缩写
    CST可以为如下4个不同的时区的缩写：
     美国中部时间：Central Standard Time (USA) UT-6:00
    澳大利亚中部时间：Central Standard Time (Australia) UT+9:30
    中国标准时间：China Standard Time UT+8:00 例如，UTC是0点，那么CST中国为早晨8点
    古巴标准时间：Cuba Standard Time UT-4:00
RTC Time
    硬件时钟时间
    set-local-rtc 1 本地时区
    set-local-rtc 0 UTC
DST Daylight Saving Time日光节约时间、夏令时
    是一种为节约能源而人为规定地方时间的制度，在这一制度实行期间所采用的统一时间称为“夏令时间”。
    一般在天亮早的夏季人为将时间提前一小时，可以使人早起早睡，减少照明量，以充分利用光照资源，从而节约照明用电。
    各个采纳夏时制的国家具体规定不同。目前全世界有近110个国家每年要实行夏令时。
    1986年至1991年，中国在全国范围实行了六年夏令时，1992年4月5日后不再实行。
```

cal

cal命令用于显示当前日历，或者指定日期的日历。

```
cal [-smjy13] [[[day] month] year]
-1 显示当月日历并将今日标黑
-3 显示上个月、当月、下个月。
-s 周日作为第一列
-m 周一作为第一列
-j 列出今天是一年的第几天
-y 列出今年所有的月
[root@rhel7 ~]# cal
      June 2016
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

[root@rhel7 ~]# cal -3
      May 2016              June 2016             July 2016
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7            1  2  3  4                  1  2
 8  9 10 11 12 13 14   5  6  7  8  9 10 11   3  4  5  6  7  8  9
15 16 17 18 19 20 21  12 13 14 15 16 17 18  10 11 12 13 14 15 16
22 23 24 25 26 27 28  19 20 21 22 23 24 25  17 18 19 20 21 22 23
29 30 31              26 27 28 29 30        24 25 26 27 28 29 30
                                            31
[root@rhel7 ~]# cal -j
         June 2016
Sun Mon Tue Wed Thu Fri Sat
            153 154 155 156
157 158 159 160 161 162 163
164 165 166 167 168 169 170
171 172 173 174 175 176 177
178 179 180 181 182
```

Linux 基础命令作业

1. 在/tmp 目录下创建 testdir 目录,在该目录下创建 file1 到 file30 的 30 个文件。
2. 删除所有以 file1 起始的文件。
3. 删除 file2 后匹配一个字符的所有文件。
4. 在/home/student 目录下递归创建一个/home/student/a/b/c/d/e 目 录,并且将该 a 目录保留原属性的复制到/tmp 目录下。
5. 将/tmp 下的 a 目录重命名为 test 目录。
6. 统计一下/etc/passwd 文件总共有几行。
7. 在/tmp 下创建一个目录,目录名为 study
8. 在 study 目录下创建一个文件, 文件名任意, 并向这个文件里写 hello 字段
9. 往该文件里追加 study 字段。
10. 查看该文件内容。
11. 统计一下该文件共有几行,几个单词,几个字节。
12. 将/etc/passwd 里以 bash 结尾的行过滤出来,并从中截取第一列字段内容。

13. 统计/boot/下所有文件的属性中含 root 字段的文件共有几个。
14. 熟悉一下 vim 编辑器,利用 vim 编辑器书写整理一下今天所学的 知识点。
15. 判断一下 boot 目录下文件的内容类型,并将属于 ASCII 码文件内容 类型的文件名显示到屏幕上