一、基础命令串讲

1. 用户组管理

1.1 用户分类

- rhel6系统：

超级用户 : 管理员root,具有所有的权限， uid=0,绝对是0

系统用户：程序用户,1 <= uid <= 499； 一般都是由其他程序建立的，用于程序或者服务运行时候的身份。不允许登录系统。

普通用户： 管理员创建的用户,500<=uid < 60000; 这些用户一般可以登录系统，对系统进行有限的维护操作。如oracle用户。

- rhel7系统：

超级用户：uid=0

系统用户：1<=uid<=999

普通用户：1000<=uid<60000

说明：

严格意义来说，系统用户中有部分用户为伪用户，伪用户一般和系统或者程序服务相关，如bin、daemon、shutdown 、halt等； 伪用户通常不需要或无法登陆系统，可以没有家目录。

重点掌握：

```
useradd
usermod
groupadd
gpasswd
chage

1. 当创建一个用时，系统会默认给该用户一些基本的东西
useradd stu1
1）/home/stu1    同时给家里面拷贝了一堆文件
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .gnome2  .mozilla

2）给了一个默认shell
/bin/sh            可以登录系统
/bin/bash          默认的shell，可以登录系统
/sbin/nologin      不能登录操作系统

3）给了uid（用户的唯一标示）和gid（用户主组）
注意：主组只能有一个，它是用户创建文件的默认所属组。

2. 用户信息保存位置
1）/etc/passwd
2）/etc/shadow   保存用户的密码信息
man 5 shadow    寻找帮助
```

```
user01:!!:17835:0:99999:7:::
```
第一列：user01(用户名)
第二列：用户加密密码。!!表示没有设置密码
第三列：17835(距离1970-01-01日最后一次修改密码的天数)；如果是0表示下次登录系统必须修改密码。
第四列:0(密码的最小生存周期)，0表示随时可以更改密码
第五列:99999（密码的最大生存周期）。如果是15表示每隔15天需要更新一次密码
第六列:7(密码过期前几天发出警告)，7代表密码过期前7天发出警告
第七列：密码的宽限期。如果是3表示，密码过期后3天依然可以访问，但是有警告；如果超过3天账号被锁，需联系管理员
第八列：账号的过期时间。
第九列：保留，未被使用

```
[root@node1 ~]# chage --help
Usage: chage [options] [LOGIN]

Options:
  -d, --lastday LAST_DAY        set date of last password change to LAST_DAY
  -E, --expiredate EXPIRE_DATE  set account expiration date to EXPIRE_DATE
  -h, --help                    display this help message and exit
  -I, --inactive INACTIVE       set password inactive after expiration
                                to INACTIVE
  -l, --list                    show account aging information
  -m, --mindays MIN_DAYS        set minimum number of days before password
                                change to MIN_DAYS
  -M, --maxdays MAX_DAYS        set maximim number of days before password
                                change to MAX_DAYS
  -W, --warndays WARN_DAYS      set expiration warning days to WARN_DAYS

[root@node1 ~]# chage -E '2018-11-01' user01
[root@node1 ~]# chage -l user01
Last password change                    : Oct 31, 2018
Password expires                        : never
Password inactive                       : never
Account expires                         : Nov 01, 2018
Minimum number of days between password change      : 0
Maximum number of days between password change      : 99999
Number of days of warning before password expires   : 7
```

3. 用户组信息
/etc/group   组基本信息
/etc/gshadow   组密码信息
man 5 gshadow

4. gpasswd命令

```
[root@node1 ~]# gpasswd --help
gpasswd: unrecognized option '--help'
Usage: gpasswd [option] GROUP

Options:
  -a, --add USER                add USER to GROUP
  -d, --delete USER             remove USER from GROUP
```

```
  -r, --remove-password         remove the GROUP's password
  -R, --restrict                restrict access to GROUP to its members
  -M, --members USER,...        set the list of members of GROUP
  -A, --administrators ADMIN,...
                                set the list of administrators for GROUP

gpasswd -a cw01 admin   将cw01添加到admin组里
gpasswd -d cw01 admin   将cw01从admin组里删除
gpasswd -M rs01,rs02 admin     设置admin组的成员列表为rs01和rs02
注意:
-M  后面所跟的用户会覆盖之前admin组里的用户
-a  往admin组追加用户
gpasswd -A rs01 admin    设置admin组管理员为rs01
注意:
管理员可以将用户剔除和添加
```

课堂实战:

1. 添加3个用户，用户harry，natasha，sarsh，要求harry，natasha用户的附加组为admin组，sarsh用户的登录shell为非交互式shell。密码均为redhat。

```
方法1:
useradd harry
useradd natasha
useradd sarsh
groupadd admin

usermod -G admin harry
usermod -G admin natasha
usermod -s /sbin/nologin sarsh
非交互式:
echo redhat|passwd --stdin harry
echo redhat|passwd --stdin natasha
echo redhat|passwd --stdin sarsh

方式2:
groupadd admin
useradd -G admin harry
useradd -G admin natasha
useradd -s /sbin/nologin sarsh

echo redhat|passwd --stdin harry
echo redhat|passwd --stdin natasha
echo redhat|passwd --stdin sarsh
```

2. 修改harry用户的家目录为/home/itcast/redhat/harry .

```
[root@node1 ~]# mkdir /home/itcast/redhat -p
[root@node1 ~]# usermod -m -d /home/itcast/redhat/harry harry
[root@node1 ~]# su - harry
[harry@node1 ~]$ pwd
/home/itcast/redhat/harry
[harry@node1 ~]$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .emacs  .gnome2  .mozilla
```

3. 修改natasha，sarsh用户的主组为itcast，并且可以登录系统.

```
[root@node1 ~]# groupadd itcast
[root@node1 ~]# usermod -g itcast sarsh
[root@node1 ~]# usermod -g itcast natasha
[root@node1 ~]# id natasha
uid=504(natasha) gid=508(itcast) groups=508(itcast),503(admin)
[root@node1 ~]# id sarsh
uid=505(sarsh) gid=508(itcast) groups=508(itcast)

[root@node1 ~]# su - sarsh
This account is currently not available.
[root@node1 ~]# usermod -s /bin/bash sarsh
[root@node1 ~]# su - sarsh
[sarsh@node1 ~]$ exit
```

4. 新建一个公司名称itcast，3个部门 cw， rs， sc；每个部门建立2个用户，如 cw01 cw02，rs01，rs02，sc01，sc02；boss01 管理公司所有部门；所有用户的密码设置为“123456”.

```
groupadd cw
groupadd rs
groupadd sc

useradd -g cw -G itcast cw01
useradd -g cw -G itcast cw02
useradd -g rs -G itcast rs01
useradd -g rs -G itcast rs02
useradd -g sc -G itcast sc01
useradd -g sc -G itcast sc02

useradd -g itcast -G cw,rs,sc boss01
```

5. 用户账号有效期3个月(90天)，第一次登录强制修改密码，每隔15天更新一次密码。

```
[root@node1 ~]# man chage
[root@node1 ~]# date -d '+90days' +%F
2019-01-29
[root@node1 ~]#
[root@node1 ~]# chage -d 0 -E '2019-01-29' -M 15 cw01
```

```
或者
[root@node1 ~]# chage -d 0 -E $(date -d '+90days' +%F) -M 15 cw01
[root@node1 ~]# chage -l cw01
Last password change                    : password must be changed
Password expires                        : password must be changed
Password inactive                       : password must be changed
Account expires                         : Jan 29, 2019
Minimum number of days between password change      : 0
Maximum number of days between password change      : 15
Number of days of warning before password expires   : 7
```

2. 权限管理

理解:

普通权限（rwx）的含义

```
权限针对的对象是文件

r: 读   4
针对文件（普通）而言，说明可以查看文件内容      cat less more head vim tail ...
针对目录而言，说明可以列出或者查看目录里的文件 ls
w: 写   2
针对文件而言，说明可以修改或者编辑文件的内容    vi vim echo ...
针对目录而言，说明可以在该目录里创建、删除、重命名等操作    touch mkdir rm mv rename...
x: 执行  1
针对文件而言，说明该文件可以执行    脚本文件、命令、程序
针对目录而言，说明可以切换到该目录里    cd

ls -l /root
drwxr-xr-x  2 root root  4096 Sep 26 15:44 aa.txt
-rw-------. 1 user01 root  1647 Mar 30  2018 anaconda-ks.cfg

stu1
uid 500 gid 500 stu1 root

user01

如何判断用户对一个文件有什么样的权限?
1. 先判断该用户是不是该文件的拥有者，如果不是再判断该文件的所属组是不是该用户的附加组
2. 可以通过步骤1来判断出该用户是不是文件的所有者后者所属组抑或是陌生人
3. 如果是文件的拥有者，那么看前三位rw-;如果是在文件的所属组里，那么看中间三位---;如果是陌生人看最后三位---


第一列(第1位):
代表文件类型    d 目录 -(f) 普通文件 l 链接文件 b 块设备 s 套接字 p 管道文件 c 字符设备
第一列（2-10位）:
文件的权限   前3位是拥有者(u)，中间三位所属组(g)，后三位其他人(o)
```

第二列:
如果是目录代表目录里的子目录个数;如果是普通的文件代表文件硬连接的个数
第三列:
文件的拥有者
第四列:
文件的所属组
第五列:
代表文件的大小,默认单位字节B, ls -lh 以人性化的方式显示
bit 0 1 1位
1字节B=8bit
1kb=1024个字节
1MB=1024kb
1GB=1024MB
1TB=1024GB
第6-8列:
文件的最后一次修改的时间
第9列:
文件名

注意:
windows下根据文件的后缀来区分文件的类型;但是在Linux下面完全不根据文件的后缀来区分。
xx.sh
xxx.tar.gz

三个特殊权限的含义

linux中除了常见的读(r)、写(w)、执行(x)权限以外,还有3个特殊的权限,分别是:
setuid、setgid和stick bit
冒险位(setuid) u+s 4000 临时拥有拥有者的权限,作用在属主上,针对命令.
chmod u+s file1
0755 4755
0644 4644
chmod u-s file1
举例:
[stu1@node1 ~]$ ll /usr/bin/vim
-rwxr-xr-x 1 root root 1967072 Apr 5 2012 /usr/bin/vim
正常情况下,stu1无法直接修改/etc/passwd文件内容

[root@node1 ~]# chmod 4755 /usr/bin/vim
或者
[root@node1 ~]# chmod u+s /usr/bin/vim
[root@node1 ~]# ll /usr/bin/vim
-rwsr-xr-x 1 root root 1967072 Apr 5 2012 /usr/bin/vim
注意:
如果原来的权限(文件拥有者那一组)上面没有x权限,那么增加完冒险位之后就会变成rwS.

强制位(setgid) g+s 2000 针对目录,任何人在该目录下创建的文件或目录都会强制继承父目录的的属组权限。
//增加强制位
chmod g+s /tmp/dir1
或者
chmod 2755 /tmp/dir1

```
[root@node1 tmp]# ll
total 4
drwxr-sr-x 2 root admin 4096 Nov  1 11:06 dir1
测试验证：
在/tmp/dir1目录创建文件，数组都属于admin
[root@node1 dir1]# ll
total 8
drwxr-sr-x 2 root admin 4096 Nov  1 11:08 aaa
drwxrwsr-x 2 stu1 admin 4096 Nov  1 11:11 bbb
-rw-r--r-- 1 root admin    0 Nov  1 11:08 file1
-rw-rw-r-- 1 stu1 admin    0 Nov  1 11:11 file2

//删除强制位
chmod g-s /tmp/dir1
chmod 0755 /tmp/dir1

黏滞位(stick bit) o+t  1000   针对公共目录，这个目录下面的文件，只有root和创建人可以删除。

chmod o+t share-dir
chmod 1755 share-dir
chmod 0755 share-dir
```

ACL访问控制策略的用途

```
setfacl
-R 递归授权，对目录下已存在的目录或文件有acl策略，但新建的文件没有
-x 去掉某个用户或者某个组的权限
-b 删除所有的acl策略
-d 递归授权(默认权限)，新建的目录会继承acl策略
mask: mask定义除other和所有人（拥有者）外的最大权限
```

```
setfacl -m u:用户:rwx  /home/redhat/file1   给单个用户单独加权限
setfacl -m g:组名:rwx  /home/redhat/file1   给单个组单独加权限

setfacl -x u:用户 /home/redhat/file1   去掉某个用户的权限
setfacl -x g:组名 filename  去掉某个组的acl策略
setfacl -x m:: filename    去掉最大acl策略
setfacl -b /home/redhat/file1   删除所有的acl策略

setfacl -m u:user01:rw file1   针对于单个用户给可读可写权限
setfacl -m g:sysadmin:rw file1  针对于单个组给可读可写权限
```

遮罩权限（默认权限）

```
用户创建创建对象的默认权限，是由umask来决定的。

目录的最大权限  0777 - umask
文件的最大权限  0666 - umask  //对于文件，永远是不能构造一个默认就能有x权限
```

临时修改root用户的umask：
umask 0033　　只针对当前用户当前终端生效
推断：root 创建的目录 744　创建文件 633 rw--wx-wx　(实际上不可能有x权限，最终权限位644)

永久生效：
1. 针对某个用户永久生效
vim ~/.bashrc
...
在文件最后增加以下行：
umask 0033

重启读取或者退出重新登录即可生效

2. 针对所有人永久生效
vim /etc/bashrc
...
umask 0033

重启读取或者退出重新登录即可生效

注意：
一般情况下，如果全局配置文件和局部配置文件冲突，以局部为准；二般情况看情况。

重点掌握：

chmod
chown
chgrp
setfacl
getfacl
针对某个人或者某个组在某个文件上增加相应权限即可。
umask
临时修改和永久修改
高级权限：
冒险位、强制位、粘滞位　　重点在于理解
　u　　g　　o

熟悉常见环境变量和shell的配置文件：

/etc/profile
系统和用户的环境变量信息，当用户第一次登录时,该文件被读取
mkdir /bin/mkdir　root　PATH=/bin
oracle　/home/oracle/bin/xxx　oracle　~/.bash_profile
/etc/bashrc
每个运行的bash信息（系统别名、函数及默认权限的定义umask），当bash shell被打开时,该文件被读取
当用户登录成功后该文件被读取

~/.bashrc
当前用户的bash信息，当用户登录和每次打开新的shell时该文件被读取
~/.bash_profile

```
当前用户的环境变量，当用户登录时，该文件被读取
~/.bash_history  记录历史命令
~/.bash_logout  当用户退出bash或者终端模式下退出登录会首先执行该文件里的代码，然后再退出。


用户登录后所读取相关环境的顺序:
/etc/profile(系统和每个用户的环境变量信息)—> ~/.bash_profile(当前用户的环境变量)—> ~/.bashrc（当前用户的bash信息）—> /etc/bashrc（每个运行的bash信息）—> ~/.bash_logout （退出bash shell时执行该文件）
```

课堂实战：

1. 使用普通用户stu1登录系统，并在/u01/STU1目录下创建一个文件zhangsan，内容为：I am jack, I want to study hard,I can do it,come on。

```
[stu1@node1 ~]$
[stu1@node1 ~]$ mkdir /u01/STU1 -p
mkdir: cannot create directory `/u01': Permission denied
[stu1@node1 ~]$ ll -d /
dr-xr-xr-x. 25 root root 4096 Nov  1 09:30 /
解决:
[root@node1 ~]# setfacl -m u:stu1:rwx
或者
[root@node1 ~]# ll /bin/mkdir
-rwxr-xr-x 1 root root 49384 Nov 22  2013 /bin/mkdir
[root@node1 ~]# chmod u+s /bin/mkdir
[root@node1 ~]# ll /bin/mkdir
-rwsr-xr-x 1 root root 49384 Nov 22  2013 /bin/mkdir
[root@node1 ~]# su - stu1
[stu1@node1 ~]$ mkdir /u01/STU1 -p
[stu1@node1 ~]$ exit
logout
[root@node1 ~]# chmod u-s /bin/mkdir
[root@node1 ~]# ll /bin/mkdir
-rwxr-xr-x 1 root root 49384 Nov 22  2013 /bin/mkdir

[root@node1 ~]# chmod g+wx -R /u01
[root@node1 ~]# su - stu1
[stu1@node1 ~]$ echo "I am jack, I want to study hard,I can do it,come on" > /u01/STU1/zhangsan
```

2. 使用stu2用户登录系统，并修改stu1用户刚刚创建的文件zhangsan，增加内容：我要和你挑战！并在相同的目录下创建一个自己的文件lisi，内容同上 I want to challenge you 。

```
[root@node1 ~]# id stu2
uid=501(stu2) gid=501(stu2) groups=501(stu2)
[root@node1 ~]# ll /u01/STU1/zhangsan
-rw-r--r-- 1 stu1 stu1 52 Nov  1 17:19 /u01/STU1/zhangsan
[root@node1 ~]# setfacl -m u:stu2:rw /u01/STU1/zhangsan
[root@node1 ~]# ll -d /u01/STU1/
drwxrwxr-- 2 root stu1 4096 Nov  1 17:19 /u01/STU1/
```

```
[root@node1 ~]# setfacl -Rm u:stu2:rwx /u01/STU1/

[root@node1 ~]# su - stu2
[stu2@node1 ~]$ cat /u01/STU1/zhangsan
I am jack, I want to study hard,I can do it,come on
[stu2@node1 ~]$ echo ".......我我我要和你挑战! " >> /u01/STU1/zhangsan
[stu2@node1 ~]$ cat>> /u01/STU1/zhangsan
I am jack, I want to study hard,I can do it,come on
我要和你挑战!
[stu2@node1 ~]$ echo " I want to challenge you 。" > /u01/STU1/lisi
```

3. stu3用户同时可以查看stu1和stu2两个用户的文件，但是不能做任何修改。

```
[root@node1 ~]# ll /u01/STU1/
total 8
-rw--w--w-  1 stu2 stu2 28 Nov  1 17:29 lisi
-rw-rw-r--+ 1 stu1 stu1 67 Nov  1 17:28 zhangsan
[root@node1 ~]# getfacl /u01/STU1/zhangsan
getfacl: Removing leading '/' from absolute path names
# file: u01/STU1/zhangsan
# owner: stu1
# group: stu1
user::rw-
user:stu2:rw-
group::r--
mask::rw-
other::r--

[root@node1 ~]# setfacl -m u:stu3:rx /u01
[root@node1 ~]# setfacl -m u:stu3:rx /u01/STU1
[root@node1 ~]# setfacl -m u:stu3:r /u01/STU1/lisi

[stu3@node1 ~]$ echo 888 >> /u01/STU1/lisi
-bash: /u01/STU1/lisi: Permission denied
[stu3@node1 ~]$ cat /u01/STU1/lisi
 I want to challenge you 。
```

3. 文件操作

重点掌握：

3.1 文件查找

```
which command     //在PATH环境变量中查找
whereis command   //查找一个命令的二进制文件，源码及手册

普通文件查找：
find  精确查找，磁盘搜索， io读写，cpu开销大
命令格式：
```

用法1：根据选项查找文件并输出到屏幕
```
find path -option 关键字
find /etc  -name '*.conf'
find /home -size +1M
find /etc -type d  -size +2M  两个条件同时满足
```

用法2：根据选项找出文件并做出相应处理
```
find path -option 关键字 [-exec 或 -ok command ] {} \;
find /tmp -mtime +3 -type f -exec rm -f {} \;
```

命令选项：
```
-name   按照文件名查找文件
-iname  按照文件名忽略大小写查找
-perm   按照文件权限来查找文件  777 744 755
-size   按照文件大小来查找
-type   按照文件类型来查找
-user   属主
-group  属组
-mtime    +n   按文件更改时间来查找文件，-n指n天以内，+n指n天以前
-atime    -n   按文件访问时间来查
-ctime     n   按文件创建时间来查找文件，-n指n天以内，+n指n天以前
```

动作：
```
-exec：对匹配的文件执行该参数所给出的shell命令。
-ok：   和-exec的作用相同，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。
-delete：删除文件
-ls：列出文件
-print：打印
```

应用：
```
1. 按照文件的修改时间、文件名、大小比较常用
-mtime   修改时间
-atime   访问时间
-ctime   属性(改变)时间 大小、权限等
2. 第二种语法用的比较多
找出来执行某个动作
-delete
-exec|-ok

3. 精确查找（多个选项一起结合使用）
-type d -a -size +1M -a -user stu1
```

示例：
```
# find /home -user jack -group hr
# find /home -user jack -a -group hr
# find /home -user jack -o -group hr
[root@node1 tmp]# find ./ -mtime -2 -a -type f -a -size -1M
./file7
./file2
./file9
./file8
./file10
```

3.2 补充扩展xargs

场景：找出某些文件将其删除或者找出某个进程将它杀死，如何实现?

```
错误:
find /path -options '关键字' |rm -f    删不成功
pgrep httpd|kill     杀不死
正确打开方式:
find /path -options '关键字' |xargs rm -f
pgrep httpd|xargs kill
```

|(管道):将上一个命令所执行的结果作为下一条命令的标准输入 **xargs:**将上一条命令所执行的结果作为下一条命令的参数

```
[root@node1 ~]# echo --help|cat
--help
[root@node1 ~]# echo --help|xargs cat


xargs:
-n: 指定单行显示的参数个数
-d: 定义分割符，默认是以空格和换行符
-i|-I: 指定替换字符，用于替换。{}
[root@node1 ~]# cat 1.txt
a b c d
10.1.1.254
A  B  C
[root@node1 ~]# cat 1.txt |xargs -n 3
a b c
d 10.1.1.254 A
B C
[root@node1 ~]# cat 1.txt |xargs -n 4
a b c d
10.1.1.254 A B C
[root@node1 ~]# cat 1.txt |xargs -d'\t' -n 3
a b c d
10.1.1.254
A B C

[root@node1 ~]# cat 1.txt |xargs -d'.' -n 3
a b c d
10 1 1
254
A  B  C


[root@node1 data]# find /data/ -type f -mtime -2|xargs -ti mv {} {}.bak
mv /data/2018-07-07.dp /data/2018-07-07.dp.bak
mv /data/2018-07-08.dp /data/2018-07-08.dp.bak
mv /data/2018-07-06.dp /data/2018-07-06.dp.bak
```

应用：
1. 理解管道和xargs的区别
2. 知道什么情况下用到xargs，xargs的一些常见的应用
找出文件执行某个命令 多个命令的结果交个xargs处理

3.3 文件打包压缩

压缩工具：
zip：兼容类unix与windows，可以压缩多个文件或目录
-r:递归压缩
用法：zip 压缩后的文件 需要压缩的文件(多个文件)
注意：
zip压缩默认压缩后的格式就是.zip；当然也可以加后缀.zip,一般都加上
解压：
unzip
-d:指定解压目录

gzip|bzip2|xz：压缩单个文件,用法相同
用法：gzip 需要压缩的文件

gzip file1 file2
gzip：压缩速度快，压缩率低，cpu开销比较低
解压：
gunzip或者gzip -d

bzip2：压缩速度慢，压缩率高，cpu开销大
压缩：
bzip2 需要压缩的文件
解压：
bzip2 需要解压的文件或者bzip2 -d 需要解压的文

xz：压缩率高，解压速度快，压缩时间较长，cpu消耗相对较大
压缩：
xz 需要压缩的文件
解压：
unxz 或者 xz -d

打包工具：
tar：打包(压缩)多个文件，不会改变文件的属性与权限
用法：
tar 选项 打包压缩后的文件 需要打包压缩的文件
tar -cvf /tmp/xxx.tar /home /etc/hosts 只打包不压缩
tar -cvzf /tmp/xxx.tar.gz /home /etc/hosts 打包并压缩
参数：
-c 打包
-z gzip
-j bzip2
-J xz
-v 显示详细信息
-f 指定包名

```
-x 解压
-r：读取追加文件    追加到tar包
-C：指定解压路径
-t:查看tar包内容

注意：
  1. 如果已经将文件压缩打包，那么就不能追加；如果只是打包就可以追加。
  2. 参数顺序需要注意，最好把-f参数放到所有参数后面。
  3. 当出现以下提示时，加一个大P参数解决。
        tar: Removing leading `/' from member names


应用：
1. tar工具，打包并压缩。语法，选项，一般-f放到所有参数最后面，结合date命令一起使用
$(date +%F)   $(date -d '+2days' +%F)
2. gzip、bzip2、xz、zip 如何压缩和解压缩，知道他们的特点
xxx.tar.gz  xxx.tar.xz  xxx.tar.bz2    tar xf
```

课堂实战：

实战1：

在/home/test目录中创建10个文件，并且修改file1到file5的时间为当前系统时间的5天前，file6的时间为9月1号，file7的时间为9月2 号。

要求：

1）找出5天以前的文件并将其删掉。

2）找出昨天的文件并将其拷贝到10.1.1.2(自己另一台主机)上的/home/test目录里，redhat用户密码为123

实战2：

1. 找出根下的所有块设备文件，并且将标准输出及标准错误重定向到/tmp/find.test文件中

```
find / -type b &>/tmp/find.test
```

2. 找出/etc/下面以.conf结尾的文件，并将其复制到/home/下的backup目录中

```
mkdir /home/backup
find /etc -name '*.conf' -type f -exec cp {} /home/backup/ \;
```

3. 将/home/backup下的所有文件全部打包压缩到/tmp/当前系统日期.tar.gz

```
cd /home/backup
tar -cvzf /tmp/$(date +%F).tar.gz ./*
```

4. 将/tmp/当前系统日期.tar.gz解压到新建目录/tmp/test中，并打包成"当前系统日期.tar"

```
mkdir /tmp/test
cd /tmp/test
tar xf /tmp/$(date +%F).tar.gz -C /tmp/test
tar cvf $(date +%F).tar  *
```

4. 进程控制

4.1 进程概述

- 什么是进程?

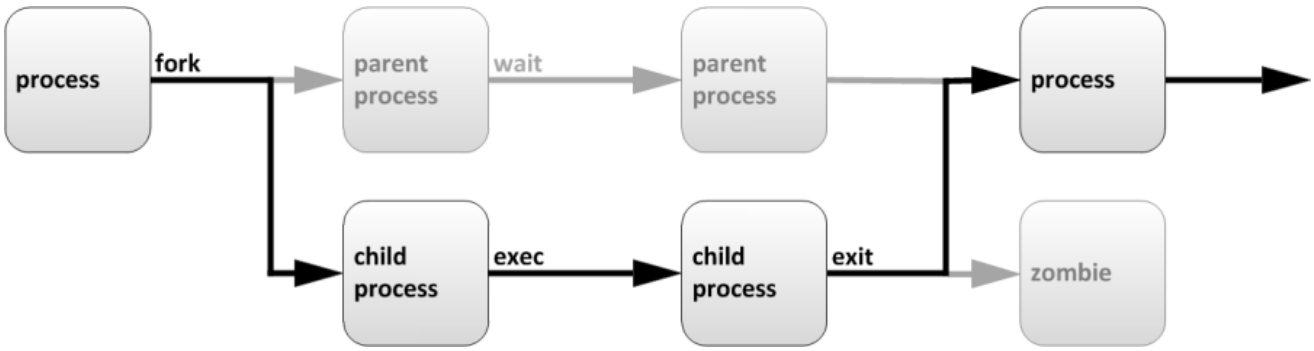
进程:是正在运行的程序，或者说是已启动的可执行程序的运行实例，进程具有自己的生命周期和各种不同的状态。

- 进程特点
  - 独立性。进程是系统中独立存在的实体，它可以拥有自己的独立资源，每一个进程都有自己的私有地址空间；在没有经过进程本身允许的情况下，一个用户进程不可以直接访问其他进程的地址空间。
  - 动态性。进程与程序的区别在于，**程序**只是一个静态的指令集合，而进程是一个正在系统中活动的指令集合；进程具有自己的生命周期和各种不同的状态。
  - 并发性。多个进程可以在单个处理器上并发执行，多个进程之间不会互相影响。
- 什么是线程?

线程：也被称作轻量级进程，线程是进程的执行单元，一个进程可以有多个线程。线程不拥有资源，它与父进程的其它线程共享该进程所拥有的资源。线程的执行是抢占式的。

- 程序和进程区别
  - 程序。二进制的文件，静态的。如:/usr/sbin/httpd，/usr/sbin/sshd
  - 进程。程序的运行过程，动态的，有生命周期及运行状态的。

4.2 进程生命周期



父进程复制自己的地址空间（fork）创建一个新的（子）进程结构。每个新进程分配一个唯一的进程 ID （PID），满足跟踪安全性之需。PID 和 父进程 ID (PPID）是子进程环境的元素，任何进程都可以创建子进程，所有进程都是第一个系统进程的后代：Centos5/6: init Centos7: systemd

子进程继承父进程的安全性身份、过去和当前的文件描述符、端口和资源特权、环境变量，以及程序代码。随后，子进程exec 自己的程序代码。通常，父进程在子进程运行期间处于睡眠（sleeping）状态。当子进程完成时发出（exit）信号请求，在退出时， 子进程会关闭或丢弃了其资源环境，剩余的部分称之为僵停（僵尸Zombie）。父进程在子进程退出时收到信号而被唤醒，清理剩余的结构，然后继续执行其自己的程序代码。

重点掌握：

静态查看进程信息ps：

```
ps aux
ps auxf：显示父子关系
ps -ef

a：显示现行终端机下的所有进程，包括其他用户的进程；
u：显示进程拥有者、状态、资源占用等的详细信息（注意有“-”和无“-”的区别）。
x：显示没有控制终端的进程。通常与 a 这个参数一起使用，可列出较完整信息。
-e：显示所有进程。
-f：完整输出显示进程之间的父子关系
-l：较长、较详细的将该 PID 的的信息列出

pidof：查看指定进程的PID
pstree：查看进程树
[root@MissHou ~]# ps aux|head
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  19356  1432 ?        Ss   19:41   0:03 /sbin/init

USER:运行进程的用户
PID:进程ID
%CPU:CPU占用率
%MEM:内存占用率
VSZ:占用虚拟内存
RSS:占用实际内存 驻留内存
TTY:进程运行的终端
STAT:进程状态     man ps (/STATE)
      R  运行
      S  可中断睡眠 Sleep
      D  不可中断睡眠
      T  停止的进程
      Z  僵尸进程
      X  死掉的进程

     Ss     s进程的领导者，父进程
     S<     <优先级较高的进程
     SN     N优先级较低的进程
     R+     +表示是前台的进程组
     Sl     以线程的方式运行

START:   进程的启动时间
TIME:     进程占用CPU的总时间
COMMAND:  进程文件，进程名
```

top命令（动态查看进程信息）：

第一部分：统计信息

```
[root@node1 ~]# top
top - 10:55:44 up  3:19,  2 users,  load average: 1.03, 0.50, 2.10
Tasks: 112 total,   1 running, 111 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.3%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   1004412k total,   331496k used,   672916k free,    58364k buffers
Swap:  2031608k total,        0k used,  2031608k free,   107896k cached
```

关注load average:系统1分钟、5分钟、15分钟内的平均负载，判断一个系统负载是否偏高需要计算单核CPU的平均负载，如下图，一般以1以内为比较合适的值。偏高说明有比较多的进程在等待使用CPU资源。

```
# From /proc/cpuinfo, system has four logical CPUs, so divide by 4:
#                              load average: 2.92, 4.48, 5.20
#           divide by number of logical CPUs:    4     4     4
#                                             ----  ----  ----
#                       per-CPU load average: 0.73  1.12  1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.
# During the last 5 minutes, the system was overloaded by ~12%.
# During the last 15 minutes, the system was overloaded by ~30%.
```

计算方法：

平均负载 / 逻辑cpu数量

```
物理CPU(N路)：主板上CPU插槽的个数
CPU核数：一块CPU上面能处理数据的芯片组的数量
逻辑CPU：一般情况，一颗cpu可以有多核，加上intel的超线程技术(HT)，可以在逻辑上再分一倍数量的cpu core出来；逻辑CPU数量=物理cpu数量 x cpu核数。如果支持HT,还要更多。

查看物理CPU的个数
#cat /proc/cpuinfo |grep "physical id"|sort |uniq|wc -l
查看逻辑CPU的个数
#cat /proc/cpuinfo |grep "processor"|wc -l
查看CPU是几核
#cat /proc/cpuinfo |grep "cores"|uniq
```

第三行：当前的CPU运行情况

```
us：用户进程占用CPU的比率
sy：内核、内核进程占用CPU的比率；
ni：如果一些用户进程修改过优先级，这里显示这些进程占用CPU时间的比率；
id：CPU空闲比率，如果系统缓慢而这个值很高，说明系统慢的原因不是CPU负载高；
wa：CPU等待执行I/O操作的时间比率，该指标可以用来排查磁盘I/O的问题，通常结合wa和id判断
hi：CPU处理硬件中断所占时间的比率；
si：CPU处理软件中断所占时间的比率；
st：其他任务所占CPU时间的比率；
说明：
1. 用户进程占比高，wa低，说明系统缓慢的原因在于进程占用大量CPU，通常还会伴有教低的id，说明CPU空闲时间很少；
2. wa低，id高，可以排除CPU资源瓶颈的可能。
3. wa高，说明I/O占用了大量的CPU时间，需要检查交换空间的使用；如果内存充足，但wa很高，说明需要检查哪个进程占用了大量的I/O资源。
```

```
在top的执行过程中，还可以使用以下的按键命令：
h|?   帮助
M  按内存的使用排序
P  按CPU使用排序
N  以PID的大小排序
R  对排序进行反转
f  自定义显示字段
1  显示所有CPU的负载
T：按该进程使用的CPU时间累积排序
k：给某个PID一个信号（signal），默认值是信号15
r：重新安排一个进程的优先级别
s：改变两次刷新之间的时间。默认是5秒
q：退出程序。

top命令常用的选项：
-d：后面可以接秒数，指定每两次屏幕信息刷新之间的时间间隔；
-p：指定某个进程来进行监控；
-b -n：以批处理方式执行top命令。通常使用数据流重定向，将处理结果输出为文件；

[root@MissHou ~]# top
[root@MissHou ~]# top -d 1
[root@MissHou ~]# top -d 1 -p 10126                查看指定进程的动态信息
[root@MissHou ~]# top -d 1 -u apache               查看指定用户的进程
[root@MissHou ~]# top -d 1 -b -n 2 > top.txt       将2次top信息写入到文件
```

4.3 进程控制

进程优先级：

```
nice，renice调整进程的优先级
```

```
-20 ~ 19  数字越低，优先级越高，系统会按照更多的cpu时间给该进程

更改现有进程的nice级别
1. 使用top更改nice级别
r  调整进程的优先级（Nice Level）  （-20高）———0———（19低）

2. 使用shell更改nice级别
[root@node1 ~]# sleep 5000 &
[1] 2544

[root@node1 ~]# renice -20 2544
2544: old priority 0, new priority -20

注意：
renice 调整一个正在运行的进程的优先级
nice  指定一个程序的优先级

启动具有不同nice级别的进程
启动进程时，通常会继承父进程的 nice级别，默认为0。
# nice -n -5 sleep 6000 &
# ps axo command,pid,nice |grep sleep
```

结束进程：

```
进程控制：
kill,killall,pkill

pkill -9 -u user
pkill -9 pts/1 -u user

给进程发送信号
# kill -l          列出所有支持的信号
编号 信号名
1) SIGHUP       重新加载配置（平滑重启）
2) SIGINT       键盘中断^C
3) SIGQUIT      键盘退出
9) SIGKILL      强制终止
15) SIGTERM     终止（正常结束），缺省信号
18) SIGCONT     继续
19) SIGSTOP     停止
20)SIGTSTP      暂停^Z


# fg  把最后放在后台运行的进程再次放到终端前台
# fg %1  将作业1调回到前台
# bg %2 把后台编号为2的进程恢复运行状态

# jobs 查看当前终端后台运行的程序
# kill -20 %3
# bg %1
```