###### 文件对比

# diff  文件名   备份文件名    //查看文件和备份文件的差别

# vimdiff    文件名  备份文件名        //在vim中显示两个文件的差异之处

tree命令

# tree a   //查看目录结构

-a 显示所有文件(包括隐藏文件)

-d 只显示目录

-L #, 显示#层级,默认显示所有的

-f 为显示完整的路径

-i 不显示树枝(目录结构的横线)

-F 在目录基名后面加上/

往一个txt文件里追加内容

    1) # echo "内容" >> txt文件

    2) # cat >> 文件绝对路径 << EOF 内容 内容 内容 内容 内容 内容 内容 内容 内容 EOF //就是把两个EOF之间的内容追加到txt文件内的内容尾,优势:可以同时追加多行.

    # for f in `seq 1000`;do touch $f.txt;done    //按序列创建1.txt ~ 1000.txt

移动命令:

# mv [-ifu] [源文件/源目录] [目标目录]

    -i 如果目标文件已经存在,询问是否覆盖

    -f 强制执行,不会询问

    -u 若目标文件已经存在,则比较文件新旧,才会移动

# rmdir 目录名    //删除空目录

###### 让test里显示除了oldboy外的其他内容.

awk sed grep

这三个命令都可以直接接文件, 不是必须接在管道符后面

grep

# grep -v oldboy test.txt    //从test.txt里过滤掉oldboy(oldboy不显示,其他内容正常显示)

sed

# sed '/oldboy/d' /tmp/test.txt    //从/tmp/test.txt里过滤掉oldboy(不显示,但其他内容正常显示)

# sed -n /oldboy/p /tmp/test.txt        //取消默认输出,打印包含oldboy的语句,相当于 grep oldboy /tmp/test.txt

# sed -n /[^oldboy]/p test.txt        //正则表达式,从/tmp/test.txt 里过滤掉oldboy, 显示其他内容

awk

# awk /[^oldboy]/ test.txt    //从/tmp/test.txt 里过滤掉oldboy, 显示其他内容

用/mnt/test.txt 覆盖掉/tmp/test,且不提示覆盖(mv 和cp 都可以)

1) # \cp -f /mnt/test.txt /tmp/test.txt    //在命令前面加上反斜线'\'

2) # /bin/cp -f /mnt/test.txt /tmp/test.txt    //用命令的全路径

只查看 ett.txt 文件内第20行到第30行的内容

# seq 100     //生成序列数字1-100

# seq -s " " 5 //-s seperator 指定分隔符,{a..z}是横着排列

➜ ~ seq -s " " 5

1 2 3 4 5

# seq 100 >ett.txt    // 在ett.txt内写入1-100的数字

方法1). # head -30 ett.txt|tail -11

方法2). # sed  -n '20,30p' ett.txt    //取消默认输出(默认输出1-100),显示20-30行,p是print

方法3). # awk '{if(NR>19 && NR<31) print $1}' ett.txt    //打印20-30行, NR是行号

find 并删除

[[root@localhost](mailto:root@localhost) tmp]# find /tmp -inum 524313 -ok rm {} \;    //-ok 可换成 -exec,-ok会在执行前询问

< rm ... /tmp/test > ? n

[[root@localhost](mailto:root@localhost) tmp]# find /tmp -inum 524313 |xargs rm    //xargs 功能和-exec一样

[[root@localhost](mailto:root@localhost) tmp]# ls

a.txt    services                                  yum\_save\_tx-2016-07-05-02-16QQh1jE.yumtx

ett.txt  yum\_save\_tx-2016-07-05-02-16nylnZA.yumtx

[[root@localhost](mailto:root@localhost) tmp]# seq -s " " 20 30      //打印,序列20-30.数字之间用" " 中间部分间隔,-s 'SEPERATOR'

20 21 22 23 24 25 26 27 28 29 30

[[root@localhost](mailto:root@localhost) tmp]#  seq -s "$" 20 30    //-s " "表示指明序列之间的间隔符号

20$21$22$23$24$25$26$27$28$29$30

[[root@localhost](mailto:root@localhost) tmp]# touch `seq -s " " 20 30`        //创建20 -30 序列文件,` `(反引号,键盘上的~键)中间可以插入命令

[[root@localhost](mailto:root@localhost) tmp]# ls

20  22  24  26  28  30     ett.txt   yum\_save\_tx-2016-07-05-02-16nylnZA.yumtx

21  23  25  27  29  a.txt  services  yum\_save\_tx-2016-07-05-02-16QQh1jE.yumtx

反选查找,就是排除a.log

[[root@localhost](mailto:root@localhost) ~]# find /tmp -type f ! -name 'a.log' -exec ls -l {} \;

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/24

-rw-------. 1 root root 209 7月   5 02:16 /tmp/yum\_save\_tx-2016-07-05-02-16nylnZA.yumtx

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/21

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/28

-rw-r--r--. 1 root root 641020 7月   5 08:46 /tmp/services

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/20

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/23

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/29

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/27

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/26

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/30

-rw-r--r--. 1 root root 12288 7月   1 12:59 /tmp/.test.swp

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/22

-rw-r--r--. 1 root root 0 7月   6 08:00 /tmp/25

-rw-r--r--. 1 root root 24576 7月   5 08:46 /tmp/.services.swp

-rw-------. 1 root root 209 7月   5 02:16 /tmp/yum\_save\_tx-2016-07-05-02-16QQh1jE.yumtx

-rw-r--r--. 1 root root 134 7月   6 02:03 /tmp/a.txt

-rw-r--r--. 1 root root 292 7月   5 06:52 /tmp/ett.txt

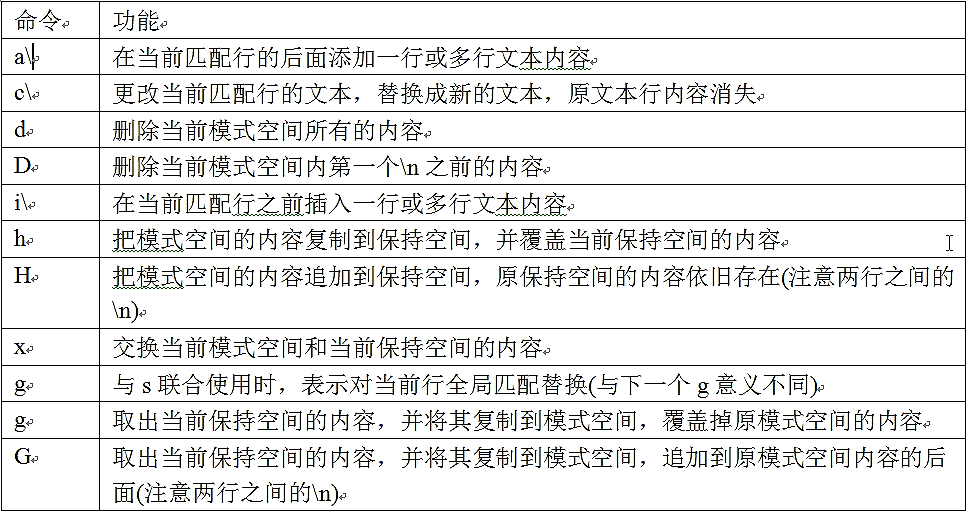
其他命令

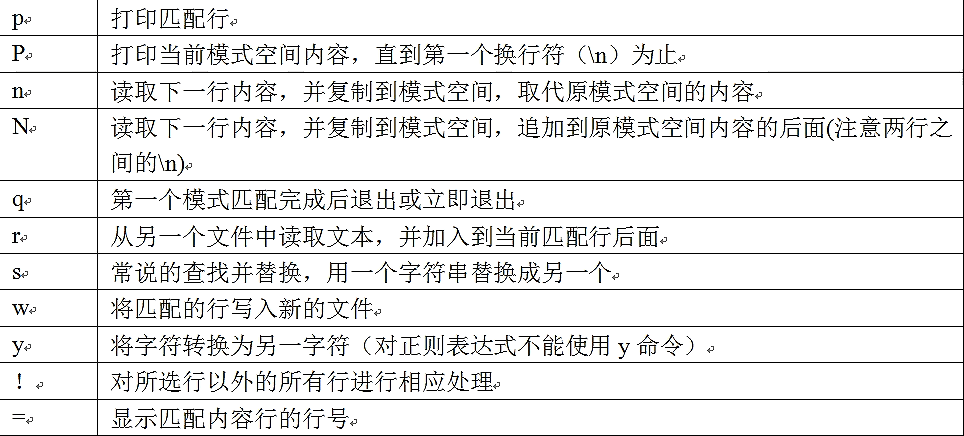
# echo "123456"|passwd --stdin aphey && history -c    // 把aphey用户密码更改为123456,一般用于批量创建用户.

# LANG=zh\_CN.UTF-8            //可以在操作界面切换语言, 中文为 zh\_CN.UTF-8; 英文为EN

# yum grouplist     //可以查看安装的软件组

###### sed 命令







###### 优化 Selinux

* 关闭Selinux

⑴ # sed -i 's/SELINUX=enforcing/SELINUX=disabled' /etc/selinux/config

也可以用 sed -i 's#SELINUX=enforcing#SELINUX=disabled#g' /etc/selinux/config

⑵ # vi /etc/selinux/config    //手动修改

    SELINUX=disabled

⑶ # setenforce=0    //临时生效,一般在配置文件里先改了,然后再用此命令

  # getenforce        //查看selinux当前状态

* 设定运行级别runlevel

    # runlevel    //查看运行级别

    # init 0-6 //重启系统并更换到对应的运行级别

# telinit 0-6 //不重启系统并更换到对应的运行级别

* Linux中文显示设置

[root@localhost]~# cat /etc/sysconfig/i18n

LANG="en\_US.UTF-8"

SYSFONT="latarcyrheb-sun16"

[root@localhost]~# cp /etc/sysconfig/i18n /etc/sysconfig.ori //备份一下

[root@localhost]~# echo 'LANG="zh\_CN.UTF-8"' > /etc/sysconfig/i18n //通过输出重定向来修改

[root@localhost]~# source /etc/sysconfig/i18n //让它生效

* 设定账号闲置超时时间

[root@localhost]~# export TMOUT=10 //超时时间为10秒,追加到/etc/profile永久生效

* 更改ssh登录标准

windows user : administrator    guest    Port:3389

Linux user; root 很多普通用户    port:22

所以我们要改掉默认的用户和端口.

# cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.bak    //先备份sshd配置文件

# vi sshd\_config    //编辑sshd\_config,常规我们纸修改下面四项

######by oldboy#2016-07-11####

Port 52113    //→ssh连接默认的端口,所有人都知道,必须要改

PermitRootLogin no    //→root用户黑客都知道,禁止它远程登录,但是可以先用普通用户登录后再换root用户

PermitEmptyPasswords no    //→禁止空密码登录

UseDNS no    //→不使用DNS

######by oldboy#2016-07-11####

# /etc/init.d/sshd restart    //→ 修改完配置文件后需要重启相关服务

# lsof -i :52113    //只知道端口,可以反查相关服务

然后从windows上用远程工具连接linux,发现连接不上

我们在windowsn上ping linuxIP, 发现可以ping通,说明网络线路是通的

然后我们用 telnet linuxIp 52113 不通, 说明sshd服务是不通的.

那我们就要到linux那去确认

1) sshd服务是否启动,用lsof -i :52113 查询即可

2) 确认防火墙是否开启 /etc/init.d/iptables status

修改/etc/ssh/sshd\_config修改端口,禁止root登录.

* history相关

1) history -c: 清空所有历史命令 -d 删除指定条目的命令 -w 保存历史命令到 ~/.bash\_history中

2) 调短历史命令的存储数量 #export HISTSIZE=5 //只保存5条命令,默认为1000

3) 控制历史记录文件(~/.bash\_history)中保存命令的长度 #export HISTFILESIZE=5

上述命令只是临时生效,要想永久生效,就把他们放到/etc/profile中,并source一下

* 克隆虚拟机网络问题处理:

1) [root@localhost]~# vi /etc/sysconfig/network-scripts/ifcfg-eth0

2) 删除掉 HWADDR行和UUID行 再清空/etc/udev/rules.d/70-persistent-net.rules来内容就可以了.

3) #reboot 重启即可

* buffer 和cache的区别?

buffer: 缓冲,从内存慢慢过渡到硬盘

cache: 缓存,从硬盘过渡到内存,别人从内存访问

* 过滤出已知当前目录下oldboy中的所有一级目录(提示:不包含oldboy目录下面目录的子目录及隐藏目录,即只能是一级目录),附上文件.

[root@localhost ~]# mkdir oldboy

[root@localhost ~]# cd oldboy

[root@localhost oldboy]# mkdir ext test xiaodong xiaofan xingfujie -p

[root@localhost oldboy]# touch ext/oldboy jeacen wodi.gz yingsui.gz

[root@localhost oldboy]# cd ..

[root@localhost ~]# tree

.

├── anaconda-ks.cfg

├── install.log

├── install.log.syslog

└── oldboy

├── ext

│   └── oldboy

├── jeacen

├── test

├── wodi.gz

├── xiaodong

├── xiaofan

├── xingfujie

└── yingsui.gz

解题思想方法:

1. 根据颜色区分文件和目录(目录的字体是浅蓝色)
2. ls -l结果中,以d开头的就是目录

[root@localhost oldboy]# ls -l | grep "^d"

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 ext

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 test

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 xiaodong

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 xiaofan

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 xingfujie

1. 通过给目录加表示,然后通过过滤标识,就可以过滤出目录[ls -F(flags不同的文件加不同的标识) 或ls -p(path,只针对目录),这两个选项就是在目录文件名后面加”/”]

[root@localhost oldboy]# ls -p | grep "/"

ext/

test/

xiaodong/

xiaofan/

xingfujie/

1. 通过find 直接查找指定类型的文件(-d 就是目录)

[root@localhost oldboy]# find ./ -maxdepth 1 -type d ! -name "."

./xingfujie

./xiaofan

./test

./xiaodong

./ext

1. ls -l输出结果中第二列数字大于1(要求文件没被硬链接)

[root@localhost oldboy]# ls -l|awk '{if($2>1)print$0}'

total 20

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 ext

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 test

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 xiaodong

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 xiaofan

drwxr-xr-x. 2 root root 4096 Jun 18 09:51 xingfujie

* 只保留7天的安全日志

//模拟环境

makedir /app/logs -p

for n in $(seq 14);

do

date -s “2016/01/$n”;

touch access\_www\_$(date +%F).log;

done

date -s “2016/01/15”

//删除7天前的文件

1. # find /app/logs -type f -mtime +7|xargs rm -r
2. # find /app/logs -type f -mtime +7-exec rm {} \;
3. # rm `find /app/logs -type f -mtime +7`

* 显示/etc/inittab文件行号的方法

1. cat -n /etc/inittab
2. nl /etc/inittab
3. awk ‘{print NR,$0}’ /etc/inittab
4. vi /etc/inittab，然后用set设置
5. grep -n “.” /etc/inittab //，是任意一个字符，前面加上行号

* 显示/proc/meminfo文件中以大小S开头的行(要求两种方式)

方法一) [root@localhost tmp]# grep "^[Ss]" /proc/meminfo

SwapCached: 0 kB

SwapTotal: 8167416 kB

SwapFree: 8167416 kB

Shmem: 3716 kB

Slab: 293272 kB

SReclaimable: 226360 kB

SUnreclaim: 66912 kB

方法二) [root@localhost tmp]# grep -i "^s" /proc/meminfo

SwapCached: 0 kB

SwapTotal: 8167416 kB

SwapFree: 8167416 kB

Shmem: 3716 kB

Slab: 293292 kB

SReclaimable: 226364 kB

SUnreclaim: 66928 kB

* 显示/etc/passwd文件中不以/bin/bash结尾的行

[root@localhost tmp]# grep -v "/bin/bash$" /etc/passwd

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

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

gopher:x:13:30:gopher:/var/gopher:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin

saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

* 显示/etc/passwd文件中ID号最大的用户的用户名

[root@localhost tmp]# sort -n -t: -k3 /etc/passwd| tail -1 | cut -d: -f1

aphey

[root@localhost tmp]# cat /etc/passwd //查看认证一下

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

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

gopher:x:13:30:gopher:/var/gopher:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin

saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

aphey:x:500:500::/home/aphey:/bin/bash

* 找出/etc/passwd中的两位或三位数

//解题思路: 需要锚定数字的位数,不锚定的话,4位数字也会被匹配进来

[root@localhost tmp]# grep "\<[[:digit:]]\{2,3\}" /etc/passwd

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

games:x:12:100:games:/usr/games:/sbin/nologin

gopher:x:13:30:gopher:/var/gopher:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

nobody:x:99:99:Nobody:/:/sbin/nologin

vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin

saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin

sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin

aphey:x:500:500::/home/aphey:/bin/bash

* 显示/etc/rc.d/rc.sysinit文件中,至少以一个空白字符开头,且后面存在非空白字符的行

[root@localhost tmp]# grep "^[[:space:]]\+" /etc/rc.d/rc.sysinit |grep "[^[:space:]]"

. /etc/sysconfig/network

HOSTNAME=localhost

mount -n -t proc /proc /proc

...

* 找出netstat -tan的结果中以"LISTEN"后跟0、1或多个空白字符结尾的行

[root@localhost tmp]# netstat -tan|grep "LISTEN[[:space:]]\*$"

tcp 0 0 127.0.0.1:25 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:7777 0.0.0.0:\* LISTEN

tcp 0 0 :::23 :::\* LISTEN

tcp 0 0 ::1:25 :::\* LISTEN

tcp 0 0 :::7777 :::\* LISTEN

* 找出/etc/passwd文件中用户名同shell名的行;**锚定单词的方法!**\_

[root@mail ~]# grep "^\([[:alnum:]]\+\>\).\*\1$" /etc/passwd

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt

* 如果root用户存在,就显示其默认的shell程序

[root@localhost tmp]# id root &>/dev/null && grep "^root" /etc/passwd|cut -d: -f7

/bin/bash

* 显示当前系统root、centos或user1的默认shell和uid

[root@mail sh]# egrep "^(root|centos|user1)\>" /etc/passwd| cut -d: -f 3,7

0:/bin/bash

* 找出/etc/rc.d/init.d/functions文件中连字符\_或者某单词后面跟一个小括号的行

[root@localhost tmp]# egrep "^[\_[:alpha:]]+\(\)" /etc/rc.d/init.d/functions

fstab\_decode\_str() {

checkpid() {

\_\_readlink() {

\_\_fgrep() {

\_\_kill\_pids\_term\_kill\_checkpids() {

\_\_kill\_pids\_term\_kill() {

\_\_umount\_loop() {

\_\_source\_netdevs\_fstab() {

\_\_source\_netdevs\_mtab() {

\_\_umount\_loopback\_loop() {

* 使用echo输出一个绝对路径，使用egrep取出其基名

[root@localhost tmp]# echo "/mnt/cdrom/images"|egrep "[^/]+/?$"

/mnt/cdrom/images

[root@localhost tmp]# echo "/mnt/cdrom/images/"|egrep "[^/]+/?$"

/mnt/cdrom/images/

// 上面两个命令的grep用-o 选项就可以显示基名

[root@localhost tmp]# echo "/mnt/cdrom/images/"|grep -E -o "[^/]+/?$"

images/

[root@localhost tmp]# echo "/mnt/cdrom/images"|grep -E -o "[^/]+/?$"

images

* 找出ifconfig命令结果中的ip地址

[root@localhost tmp]# ifconfig| awk 'NR==2{print}'| awk -F "[ :]+" '{print $4}'

192.168.88.123

* 优化开机启动

// 1) 关闭不需要的启动项;

[root@localhost ~]# chkconfig --list|grep "3:on"| grep -Ev "crond|sshd|rsyslog|network|sysstat"| awk '{print "chkconfig",$1,”off”}'|bash

思路: 列出所有开机启动的项目, 选出除了我不需要的项,通过awk 打印出命令,再交给bash执行

// 思路2) 全部关闭,开启我们需要的启动项

[root@localhost ~]# chkconfig --list| grep "3:on"|awk '{print "chkconfig",$1,"off"}'|bash //把chkconfig中再三级别自动开启的服务全部关闭

[root@localhost ~]# chkconfig --list| grep "3:on" //已经全关了

[root@localhost ~]# chkconfig --list| grep "3:off" //查看开机启动关闭的项

auditd 0:off 1:off 2:off 3:off 4:off 5:off 6:off

blk-availability 0:off 1:on 2:off 3:off 4:off 5:off 6:off

crond 0:off 1:off 2:off 3:off 4:off 5:off 6:off

ip6tables 0:off 1:off 2:off 3:off 4:off 5:off 6:off

iptables 0:off 1:off 2:off 3:off 4:off 5:off 6:off

lvm2-monitor 0:off 1:on 2:off 3:off 4:off 5:off 6:off

netconsole 0:off 1:off 2:off 3:off 4:off 5:off 6:off

netfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off

network 0:off 1:off 2:off 3:off 4:off 5:off 6:off

postfix 0:off 1:off 2:off 3:off 4:off 5:off 6:off

rdisc 0:off 1:off 2:off 3:off 4:off 5:off 6:off

restorecond 0:off 1:off 2:off 3:off 4:off 5:off 6:off

rsyslog 0:off 1:off 2:off 3:off 4:off 5:off 6:off

saslauthd 0:off 1:off 2:off 3:off 4:off 5:off 6:off

sshd 0:off 1:off 2:off 3:off 4:off 5:off 6:off

sysstat 0:off 1:on 2:off 3:off 4:off 5:off 6:off

udev-post 0:off 1:on 2:off 3:off 4:off 5:off 6:off

xinetd 0:off 1:off 2:off 3:off 4:off 5:off 6:off

[root@localhost ~]# chkconfig --list| grep "3:off"|egrep "crond|sshd|rsyslog|network|sysstat" // 选出我们要自启动的项

crond 0:off 1:off 2:off 3:off 4:off 5:off 6:off

network 0:off 1:off 2:off 3:off 4:off 5:off 6:off

rsyslog 0:off 1:off 2:off 3:off 4:off 5:off 6:off

sshd 0:off 1:off 2:off 3:off 4:off 5:off 6:off

sysstat 0:off 1:on 2:off 3:off 4:off 5:off 6:off

//下面这条命令就是将我们要的5项自启动开启起来

[root@localhost ~]# chkconfig --list| grep "3:off"|egrep "crond|sshd|rsyslog|network|sysstat"|awk '{print "chkconfig",$1,"on"}'|bash

[root@localhost ~]# chkconfig --list| grep "3:on" //成功

crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off

network 0:off 1:off 2:on 3:on 4:on 5:on 6:off

rsyslog 0:off 1:off 2:on 3:on 4:on 5:on 6:off

sshd 0:off 1:off 2:on 3:on 4:on 5:on 6:off

sysstat 0:off 1:on 2:on 3:on 4:on 5:on 6:off

* find命令中的{}是前向引用的意思,指代前面的搜索结果

# aphey @ training in ~/test [16:12:40]

$ find . -type f -iname "file1.txt" -exec cp {} /tmp \;

// 上面的例子中的{}就是指代前面的额file1.txt

# aphey @ training in ~/test [16:13:26]

$ ls /tmp

file1.txt hello inittab num services test

# aphey @ training in ~/test [16:17:02] C:123

$ find . -type f -iname "file2.txt"|xargs cp {} /tmp

cp: 目标"./file2.txt" 不是目录

# aphey @ training in ~/test [16:17:24] C:123

$ find . -type f -iname "file2.txt"|xargs -i cp {} /tmp

* xargs的i选项

xargs与find经常结合来进行文件操作，平时删日志的时候只是习惯的去删除，比如

# find . -type f -name "\*.log" | xargs rm -rf \*

就将以log结尾的文件删除了，如果我想去移动或者复制就需要使用参数来代替了。

xargs -i 参数或者-I参数配合{}即可进行文件的操作。

man了一下看的还是不太懂，通过例子，做作实验将我的理解写一下。

############### 操作的目录下的文件###############

[root@test05 ab]# ls

1kk.zip 3kk.zip 5kk.zip b.rar d.rar f.rar h.rar j.rar mini.txt ni.txt

2kk.zip 4kk.zip a.rar c.rar e.rar g.rar i.rar k.rar nii.txt

###################使用 i 参数 ##################

[root@test05 ab]# find . -type f -name "\*.txt" | xargs -i cp {} /tmp/k/

[root@test05 ab]# ls ../k/

mini.txt nii.txt ni.txt

[root@test05 ab]#

################### 使用 I 参数 ################

[root@test05 ab]# find . -type f -name "\*.txt" | xargs -I{} cp {} /tmp/n/

[root@test05 ab]# ls ../n/

mini.txt nii.txt ni.txt

结果出来了，

加-i 参数直接用 {}就能代替管道之前的标准输出的内容；

加 -I 需要事先指定替换字符.

* xargs命令,可以从标准输入(管道或stdin)获取参数,并将参数转换成其他命令的参数

# aphey @ training in ~/test [16:42:24]

$ cat >>test<<EOF

heredoc> 1 2 3 4 5 6

heredoc> 7 8 9

heredoc> 10 11

heredoc> EOF

# aphey @ training in ~/test [16:42:42]

$ cat test

1 2 3 4 5 6

7 8 9

10 11

# aphey @ training in ~/test [16:42:46]

$ xargs < test //输入重定向,把多行输入变成单行输出

1 2 3 4 5 6 7 8 9 10 11

// xargs -n # 选项,设定每行的最大参数的个数

# aphey @ training in ~/test [16:44:16]

$ xargs -n 4 <test

1 2 3 4

5 6 7 8

9 10 11

// xargs -i选项,指明命令后面的{}是代替前面的参数, -I 选项则需要用-I{} COMMD {}格式

# aphey @ training in ~/test [16:47:30]

$ ls

dir1 dir2 dir3 file1.txt file2.txt file3.txt file4.txt file5.txt test

# aphey @ training in ~/test [16:50:45]

$ find . -type f -iname "file1.txt"|xargs -i cp {} /tmp

# aphey @ training in ~/test [16:51:24]

$ ls /tmp

file1.txt file2.txt hello inittab num services test

// -I{} 选项

# aphey @ training in ~/test [16:52:30] C:127

$ find . -type f -iname "file4.txt"|xargs -I{} cp {} /tmp

# aphey @ training in ~/test [16:52:35]

$ ls /tmp

file1.txt file2.txt file4.txt hello inittab num services test

* 查看文件的inode

// 方法一) 用ls -li FILE\_NAME

# aphey @ training in ~/test [15:34:22]

$ ls -li file1.txt

1835929 -rw-r--r--. 1 root root 0 6月 19 08:30 file1.txt

// 方法二) 用stat FILE\_NAME

# aphey @ training in ~/test [15:36:03] C:1

$ stat file1.txt

File: "file1.txt"

Size: 0 Blocks: 0 IO Block: 4096 普通空文件

Device: fd02h/64770d Inode: 1835929 Links: 1

Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)

Access: 2018-06-19 16:13:26.816857586 +0800

Modify: 2018-06-19 08:30:05.077895888 +0800

Change: 2018-06-19 10:44:31.554886008 +0800

* 查看文件的inode的大小

# aphey @ training in ~/test [13:25:45]

$ sudo dumpe2fs /dev/sda1|grep -i "inode size"

dumpe2fs 1.41.12 (17-May-2010)

Inode size: 128

* 查看每个分区inode的数量和使用量

# aphey @ training in ~/test [13:26:27] C:1

$ df -i /dev/sda1

Filesystem Inodes IUsed IFree IUse% Mounted on

/dev/sda1 128016 44 127972 1% /boot

* 查看Inode的数量和Block的数量

# aphey @ training in ~/test [15:32:07] C:1

$ sudo dumpe2fs /dev/sda1|egrep -i "block count|inode count"

[sudo] password for aphey:

dumpe2fs 1.41.12 (17-May-2010)

Inode count: 128016

Block count: 512000

Reserved block count: 25600

* 面试题: 如果向磁盘写入数据提示: No space left on device,通过df-h查看磁盘空间,发现没满,请问可能是什么导致的?

小文件(文件大小小与block),inode被耗尽了; 企业工作中邮件临时队列/var/spool/clientmqueue/这里很容易被大量小文件沾满导致No space left on device的错误,clientmqueue目录只有安装了sendmail服务才会有,是sendmail邮件的临时队列,centos5.8默认就会装,CentOS6默认没安装sendmail,但是有postfix.

* 查看文件类型的命令.

# aphey @ training in ~/test [16:40:47]

$ file /var/log/lastlog

/var/log/lastlog: data

# aphey @ training in ~/test [16:41:00]

$ file /bin/bash

/bin/bash: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped(executable and linkable Format)

* readlink命令,根据软连接文件查找源文件的名字

# aphey @ training in ~/test [11:00:05]

$ touch file

# aphey @ training in ~/test [11:00:34] C:1

$ ln -s file ./file\_soft

# aphey @ training in ~/test [11:01:06] C:1

$ readlink file\_soft

file

* 通过mkdir命令创建一个新目录/oldboy/ett,它的硬链接数是多少,为什么?如果在/olboy/ett下面在创建一个目录test.再问,/oldboy/ett的硬链接数是多少?为什么?

➜ ~ mkdir /oldboy/ett

➜ ~ ll -di /oldboy/ett //有两个,分别是自己和目录下的.

1179654 drwxr-xr-x. 2 root root 4.0K Jun 21 14:40 /oldboy/ett

➜ ~ ll -di /oldboy/ett /oldboy/ett/.

1179654 drwxr-xr-x. 2 root root 4.0K Jun 21 14:40 /oldboy/ett

1179654 drwxr-xr-x. 2 root root 4.0K Jun 21 14:40 /oldboy/ett/.

//如果在/oldboy/ett/下再创建一个test,那么/oldboy/ett硬链接数就会变成3个.

➜ ~ ll -di /oldboy/ett /oldboy/ett/. /oldboy/ett/test/..

1179654 drwxr-xr-x. 3 root root 4.0K Jun 21 14:49 /oldboy/ett

1179654 drwxr-xr-x. 3 root root 4.0K Jun 21 14:49 /oldboy/ett/.

1179654 drwxr-xr-x. 3 root root 4.0K Jun 21 14:49 /oldboy/ett/test/..

// 在父目录里创建一个子目录,父目录的硬链接就会增加1;目录是不能通过ln命令来创建硬链接的

* linux删除文件的原理

文件的删除由两个控制:

i\_link:文件的硬链接数

i\_count: 进程的引用基数

//静态文件(没有被进程调用的文件)

当某个文件及它的硬链接都被删除后,它的inode 会被系统回收.这个文件就被彻底删除了

//动态文件:

当文件的i\_link和i\_count都为0的时候,文件才是被删除