

Linux 学习笔记

一、常用命令

LINUX常用操作命令和命令行编辑快捷键

终端快捷键:

Ctrl + a/Home 切换到命令行开始

Ctrl + e/End 切换到命令行末尾

Ctrl + l 清除屏幕内容, 效果等同于 clear

Ctrl + u 清除剪切光标之前的内容

Ctrl + k 剪切清除光标之后的内容

Ctrl + y 粘贴刚才所删除的字符

Ctrl + r 在历史命令中查找 (这个非常好用, 输入关键字就调出以前的命令了)

Ctrl + c 终止命令

ctrl + o 重复执行命令

Ctrl + d 退出 shell, logout

Ctrl + z 转入后台运行,但在当前用户退出后就会终止

Ctrl + t 颠倒光标所在处及其之前的字符位置, 并将光标移动到下一个字符

Alt + t 交换当前与以前单词的位置

Alt + d 剪切光标之后的词

Ctrl+w 剪切光标所在处之前的一个词 (以空格、标点等为分隔符)

Ctrl+ (x u) 按住 Ctrl 的同时再先后按 x 和 u, 撤销刚才的操作

Ctrl+s 锁住终端

Ctrl+q 解锁终端

!! 重复执行最后一条命令

history 显示你所有执行过的编号+历史命令。这个可以配合!编辑来执行某某命令

!\$ 显示系统最近的一条参数

最后这个比较有用, 比如我先用 cat /etc/sysconfig/network-scripts/ifconfig-eth0, 然后我想用 vim 编辑。

一般的做法是先用 ↑ 显示最后一条命令, 然后用 Home 移动到命令最前, 删除 cat, 然后再输入 vim 命令。其实完全可以用 vim !\$来代替。

gnome快捷键

Alt + F1 类似 Windows 下的 Win 键, 在 GNOME 中打开"应用程序"菜单(Applications)

Alt + F2 类似 Windows 下的 Win + R 组合键, 在 GNOME 中运行应用程序

Ctrl + Alt + D 类似 Windows 下的 Win + D 组合键, 显示桌面

Ctrl + Alt + L 锁定桌面并启动屏幕保护程序

Alt + Tab 同 Windows 下的 Alt + Tab 组合键, 在不同程序窗口间切换

PrintScreen 全屏抓图

Alt + PrintScreen 当前窗口抓图

Ctrl + Alt + → / ← 在不同工作台间切换

Ctrl + Alt + Shift + → / ← 移动当前窗口到不同工作台

Ctrl+Alt+Fn 终端 N 或模拟终端 N(n 和 N 为数字 1-6)

Ctrl+Alt+F7 返回桌面

窗口操作快捷键

Alt + F4 关闭窗口

Alt + F5 取消最大化窗口 (恢复窗口原来的大小)

Alt + F7 移动窗口 (注: 在窗口最大化的状态下无效)

Alt + F8 改变窗口大小 (注: 在窗口最大化的状态下无效)

Alt + F9 最小化窗口

Alt + F10 最大化窗口

Alt + 空格键 打开窗口的控制菜单 (点击窗口左上角图标出现的菜单)

文件浏览器

Ctrl+N 新建窗口

Ctrl + Shift + W 关闭所有文件浏览器

Ctrl + 1/2 改变文件夹视图查看方式, 图标视图/列表视图

Alt + → / ← 后退/前进

Alt + ↑ / ↓ 移动到父文件夹/选择的文件夹

Alt + Home 直接移动到主文件夹

F9 开关显示隐藏 Nautilus 侧边栏

Ctrl+H 显示隐藏文件 (切换键)

Shift+Ctrl+N 新建文件夹, 很有用

Alt + Enter 查看选择文件/文件夹的属性, 代替单击右键选择属性

Ctrl+Page Up 上一个标签

Ctrl+Page Down 下一个标签

Alt+N 切换到第 N 个标签 (N 为数字)

关机和重启命令

Shutdown

Reboot

Halt

poweroff

grep和管道符

昨天的时候 leader 给我出了道问题:

找出文件夹下包含 “aaa” 同时不包含 “bbb” 的文件, 然后把他们重新生成一下。要求只能用一行命令。

我对 Linux 是个白痴, 工作了之后才开始接触的, 会用的命令只有那几个常用的。这个问题对我来说就有点难度, 我只是大概知道查找文件用 **grep**, 其他的就不知道了。不过没关系, 用 **Google**, 查找到 **grep** 的完整用法:

- 1、**grep -l 'boss' *** 显示所有包含 boss 的文件名。
- 2、**grep -n 'boss' file** 在匹配行之前加行号。
- 3、**grep -i 'boss' file** 显示匹配行, boss 不区分大小写。
- 4、**grep -v 'boss' file** 显示所有不匹配行。
- 5、**grep -q 'boss' file** 找到匹配行, 但不显示, 但可以检查 **grep** 的退出状态。(0 为匹配成功)
- 6、**grep -c 'boss' file** 只显示匹配行数 (包括 0)。

7、grep "\$boss" file 扩展变量 boss 的值再执行命令。

8、ps -ef|grep "^*user1" 搜索 user1 的命令，即使它前面有零个或多个空格。

9、ps -e|grep -E 'grant_server|commsvr|tcpsvr|dainfo' 查找多个字符串的匹配（grep -E 相当于 egrep）

了解了 grep 的参数之后，问题就解决了一半了，因为可以搜索出符合条件的文件了。不过光有 grep 还是不行，因为要把搜索出来的文件名作为参数传给 generate 命令。OK，接下来该管道符出场了。

即使是像我这样对 Linux 只是有一点了解的人也经常用到管道符，比如“|”，示例：ls -a | more 。但是对于管道符的具体意义和它做了什么我就知道了，没关系，Google 一下，找到一些资料：

利用 Linux 所提供的管道符“|”将两个命令隔开，管道符左边命令的输出就会作为管道符右边命令的输入。连续使用管道意味着第一个命令的输出会作为第二个命令的输入，第二个命令的输出又会作为第三个命令的输入，依此类推。

所以查找的时候可以这样写：

```
grep -rl "aaa" * | grep -v "bbb"
```

这样右边的命令就可以从前面的结果中筛选了。然后还有 generate 命令，因为生成文件的命令格式是这样的：

generate 文件名

不过如果直接使用 generate grep -rl "aaa" * | grep -v "bbb" 的话会出错，因为命令会从左向右执行，这条命令就会把 grep 作为一个文件名来看待。怎么办呢？这个时候就要使用 · (键盘上数字键 1 旁边的那个符号，和“~”在一个按键上)来做命令替换了，用 · 把后面的 grep 命令包起来就好了，这样：

```
generate ·grep -rl "aaa" * | grep -v "bbb"·
```

然后就搞定了。

工作一段时间之后，越来越喜欢 Linux 的哲学了，它有很多命令，看起来功能都不是那么强劲，但是如果你开动脑筋把这些命令组合起来的话，就能实现 很多让你意想不到的功能，有时候你忍不住惊呼：实在

是太 coooooool 了！这对于像我这种被 Windows 的傻瓜式操作惯坏了的人来说，是个福音，以后要多多开动生锈了的大脑。如果单纯使用电脑的话，还是 Windows 好用，但是对于程序员，最好还是多玩玩 Linux。

BTW，现在也越来越喜欢使用 VIM 了，虽然刚开始用的时候就觉得它是个记事本~囧~ 以前总听说“真正的牛人编码都是用记事本编写的”，当时就觉得这些人实在太厉害了，代码提示和自动补全都不用，现在想想，可能是外行看到他们使用灵活+强大的 VIM 或者 EMACS 了吧。^_^

我的补充：

查找包含 logFileId 又包含 open 的文件：

```
用 grep "logFileId" *.tbc|grep "open"
```

二、磁盘管理

文件系统配置文件

/etc/filesystems: 系统指定的测试挂载文件系统类型

/proc/filesystems: Linux 系统已经加载的文件系统类型

/lib/modules/2.6.18-274.el5/kernel/fs/ 文件系统类型的驱动所在目录

/etc/fstab

/etc/mtab

linux 文件类型的颜色

linux 文件颜色的含义: 蓝色代表目录 绿色代表可执行文件 红色表示压缩文件 浅蓝色表示链接文件 灰色表示其他文件 红色闪烁表示链接的文件有问题了 黄色表示设备文件

蓝色文件-----目录

白色文件-----一般性文件，如文本文件，配置文件，源码文件等

浅蓝色文件-----链接文件，主要是使用 ln 命令建立的文件

绿色文件-----可执行文件，可执行的程序

红色文件-----压缩文件或者包文件

Linux 下用字符表示的文件类型

-: 普通文件

d: 目录文件

l: 链接文件

b: 块设备文件

c: 字符设备文件

p: 管道文件

文件系统操作命令

df: 列出文件系统的整体磁盘使用情况

```
[root@centos57 ~]# df -h
```

文件系统	容量	已用	可用	已用%	挂载点
/dev/mapper/VolGroup00-LogVol00	16G	4.2G	11G	28%	/
/dev/sda1	99M	13M	81M	14%	/boot
tmpfs	1005M	0	1005M	0%	/dev/shm

```
[root@centos57 ~]# df -i
```

文件系统	Inode (I)已用	(I)可用	(I)已用%	挂载点	
/dev/mapper/VolGroup00-LogVol00	4186112	154441	4031671	4%	/
/dev/sda1	26104	36	26068	1%	/boot
tmpfs	257210	1	257209	1%	/dev/shm
.host:/	0	0	0	-	/mnt/hgfs

du: 列出目录所占空间

du -sh 显示当前目录大小

du -sh / 显示/目录下的所有目录大小

dumpe2fs: 显示当前的磁盘状态

ln: 连接文件（快捷方式）

ln -sf 源文件 目标文件

不加任何参数就进行连接，就是 hard link，加上-s 就是 Symbolic link，hard link 不支持目录和跨文件系统。

Fdisk

Fdisk 不支持大于 2T 的磁盘

Fdisk -l 显示系统中的所有分区内容

```
[root@centos57 ~]# fdisk -l
```

Disk /dev/sda: 21.4 GB, 21474836480 bytes

255 heads, 63 sectors/track, 2610 cylinders

总扇区数，可以和下面的最后扇区数比较，看剩余

Units = cylinders of 16065 * 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	13	104391	83	Linux
/dev/sda2		14	2610	20860402+	8e	Linux LVM

[root@centos57 ~]# fdisk /dev/sda2

Parted: 2T以上磁盘分区工具

支持大于 2T 的磁盘，2T 以下的最好还是用 Fdisk 来分区。

[root@centos57 aixi]# parted /dev/hda print

Model: VMware Virtual IDE Hard Drive (ide)

Disk /dev/hda: 2147MB

Sector size (logical/physical): 512B/512B

Partition Table: msdos

Number	Start	End	Size	Type	File system	标志
1	32.3kB	101MB	101MB	主分区	ext3	
2	101MB	357MB	256MB	主分区	linux-swap	

parted /dev/hda rm 2 删除第 2 个分区

parted /dev/hda mkpart primary ext3 120MB 200MB 创建分区，primary 代表主分区，还可以是 extended 扩展分区，logical 逻辑分区;ext3 代表分区类型，120MB 是开始位置，最好是接上一分区的结束位置，200M 是结束位置

partprobe :更新分区表/磁盘

用于重读分区表，当出现删除文件后,出现仍然占用空间。可以 partprobe 在不重启的情况下重读分区

partprobe

这个命令执行完毕之后不会输出任何返回信息，你可以使用 mke2fs 命令在新的分区上创建文件系统。

Mkfs:磁盘格式化

Mkfs -t 文件系统格式 设备文件名（盘符）

[root@centos57 ~]# mkfs -t ext3 /dev/hda1

e2label: 设置磁盘卷标

e2label 设备名称 新 label 名称，可以用 dumpe2fs 查看卷标

[root@centos57 ~]# e2label /dev/hda1 aixi

Mount:挂载磁盘

命令格式：

mount [-t vfstype] [-o options] device dir

mount -o remount,rw,auto / 重新挂载

mount -n -o remount,rw / 重新挂载根目录，设置为可读写

其中：

1.-t vfstype 指定文件系统的类型，通常不必指定。mount 会自动选择正确的类型。常用类型有：

光盘或光盘镜像：iso9660

DOS fat16 文件系统：msdos

Windows 9x fat32 文件系统：vfat

Windows NT ntfs 文件系统：ntfs

Mount Windows 文件网络共享：smbfs

UNIX(LINUX) 文件网络共享: nfs

2.-o options 主要用来描述设备或档案的挂载方式。常用的参数有:

loop: 用来把一个文件当成硬盘分区挂载上系统

ro: 采用只读方式挂载设备

rw: 采用读写方式挂载设备

iocharset: 指定访问文件系统所用字符集

3.device 要挂载(mount)的设备。

4.dir 设备在系统上的挂载点(mount point)。

挂载光盘镜像文件

1、从光盘制作光盘镜像文件。将光盘放入光驱, 执行下面的命令。

```
#cp /dev/cdrom /home/sunky/mydisk.iso 或
```

```
#dd if=/dev/cdrom of=/home/sunky/mydisk.iso
```

注: 执行上面的任何一条命令都可将当前光驱里的光盘制作成光盘镜像文件/home/sunky/mydisk.iso

2、将文件和目录制作成光盘镜像文件, 执行下面的命令。

```
#mkisofs -r -J -V mydisk -o /home/sunky/mydisk.iso /home/sunky/ mydir
```

注: 这条命令将/home/sunky/mydir 目录下所有的目录和文件制作成光盘镜像文件

/home/sunky/mydisk.iso, 光盘卷标为: mydisk

3、光盘镜像文件的挂载(mount)

```
#mkdir /mnt/vcdrom
```

注: 建立一个目录用来作挂载点(mount point)

```
#mount -o loop -t iso9660 /home/sunky/mydisk.iso /mnt/vcdrom
```

注: 使用/mnt/vcdrom 就可以访问盘镜像文件 mydisk.iso 里的所有文件了。

挂载移动硬盘

对 linux 系统而言, USB 接口的移动硬盘是当作 SCSI 设备对待的。插入移动硬盘之前, 应先用 fdisk -l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况。

```
[root at pldyrouter /]# fdisk -l
```

```
Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes
```

```
255 heads, 63 sectors/track, 8924 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes
```

```
Device Boot Start End Blocks Id System
```

```
/dev/sda1 1 4 32098+ de Dell Utility
```

```
/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS
```

```
/dev/sda3 2555 7904 42973875 83 Linux
```

```
/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)
```

```
/dev/sda5 7905 8924 8193118+ 82 Linux swap
```

在这里可以清楚地看到系统有一块 SCSI 硬盘/dev/sda 和它的四个磁盘分区/dev/sda1 -- /dev/sda4, /dev/sda5 是分区/dev/sda4 的逻辑分区。接好移动硬盘后, 再用 fdisk -l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况

```
[root at pldyrouter /]# fdisk -l
```

```
Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes
```

```
255 heads, 63 sectors/track, 8924 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes
```

```

Device Boot Start End Blocks Id System
/dev/sda1 1 4 32098+ de Dell Utility
/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS
/dev/sda3 2555 7904 42973875 83 Linux
/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)
/dev/sda5 7905 8924 8193118+ 82 Linux swap
Disk /dev/sdc: 40.0 GB, 40007761920 bytes
255 heads, 63 sectors/track, 4864 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sdc1 1 510 4096543+ 7 HPFS/NTFS
/dev/sdc2 511 4864 34973505 f Win95 Ext'd (LBA)
/dev/sdc5 511 4864 34973473+ b Win95 FAT32

```

大家应该可以发现多了一个 SCSI 硬盘/dev/sdc 和它的两个磁盘分区/dev/sdc1?、/dev/sdc2,其中/dev/sdc5 是/dev/sdc2 分区的逻辑分区。我们可以使用下面的命令挂接/dev/sdc1 和/dev/sdc5。

```

#mkdir -p /mnt/usbhd1
#mkdir -p /mnt/usbhd2
注：建立目录用来作挂接点(mount point)
#mount -t ntfs /dev/sdc1 /mnt/usbhd1
#mount -t vfat /dev/sdc5 /mnt/usbhd2

```

注：对 ntfs 格式的磁盘分区应使用-t ntfs 参数，对 fat32 格式的磁盘分区应使用-t vfat 参数。若汉字文件名显示为乱码或不显示，可以使用下面的命令格式。

```

#mount -t ntfs -o iocharset=cp936 /dev/sdc1 /mnt/usbhd1
#mount -t vfat -o iocharset=cp936 /dev/sdc5 /mnt/usbhd2

```

linux 系统下使用 fdisk 分区命令和 mkfs 文件系统创建命令可以将移动硬盘的分区制作成 linux 系统所特有的 ext2、ext3 格式。这样，在 linux 下使用就更方便了。使用下面的命令直接挂接即可。

```

#mount /dev/sdc1 /mnt/usbhd1

```

挂接U盘

和 USB 接口的移动硬盘一样对 linux 系统而言 U 盘也是当作 SCSI 设备对待的。使用方法和移动硬盘完全一样。插入 U 盘之前，应先用 fdisk -l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况。

```

[root at pldyrouter root]# fdisk -l
Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes
255 heads, 63 sectors/track, 8924 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sda1 1 4 32098+ de Dell Utility
/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS
/dev/sda3 2555 7904 42973875 83 Linux
/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)
/dev/sda5 7905 8924 8193118+ 82 Linux swap

```

插入 U 盘后，再用 fdisk -l 或 more /proc/partitions 查看系统的硬盘和硬盘分区情况。

```

[root at pldyrouter root]# fdisk -l
Disk /dev/sda: 73 dot 4 GB, 73407820800 bytes

```

255 heads, 63 sectors/track, 8924 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sda1 1 4 32098+ de Dell Utility
/dev/sda2 * 5 2554 20482875 7 HPFS/NTFS
/dev/sda3 2555 7904 42973875 83 Linux
/dev/sda4 7905 8924 8193150 f Win95 Ext'd (LBA)
/dev/sda5 7905 8924 8193118+ 82 Linux swap

Disk /dev/sdd: 131 MB, 131072000 bytes

9 heads, 32 sectors/track, 888 cylinders

Units = cylinders of 288 * 512 = 147456 bytes

Device Boot Start End Blocks Id System

/dev/sdd1 * 1 889 127983+ b Win95 FAT32

Partition 1 has different physical/logical endings:

phys=(1000, 8, 32) logical=(888, 7, 31)

系统多了一个 SCSI 硬盘/dev/sdd 和一个磁盘分区/dev/sdd1,/dev/sdd1 就是我们要挂载的 U 盘。

#mkdir -p /mnt/usb

注：建立一个目录用来作挂载点(mount point)

#mount -t vfat /dev/sdd1 /mnt/usb

注：现在可以通过/mnt/usb 来访问 U 盘了，若汉字文件名显示为乱码或不显示，可以使用下面的命令。

#mount -t vfat -o iocharset=cp936 /dev/sdd1 /mnt/usb

挂载Windows文件共享

Windows 网络共享的核心是 SMB/CIFS，在 linux 下要挂载(mount)windows 的磁盘共享，就必须安装和使用 samba 软件包。现在流行的 linux 发行版绝大多数已经包含了 samba 软件包，如果安装 linux 系统时未安装 samba 请首先安装 samba。当然也可以到 www.samba.org 网站下载.....新的版本是 3.0.10 版。

当 windows 系统共享设置好以后，就可以在 linux 客户端挂载(mount)了，具体操作如下：

mkdir -p /mnt/samba

注：建立一个目录用来作挂载点(mount point)

mount -t smbfs -o username=administrator,password=pldy123 //10.140.133.23/c\$ /mnt/samba

注：administrator 和 pldy123 是 ip 地址为 10.140.133.23 windows 计算机的一个用户名和密码，c\$是这台计算机的一个磁盘共享

如此就可以在 linux 系统上通过/mnt/samba 来访问 windows 系统磁盘上的文件了。以上操作在 redhat as server 3、redflag server 4.1、suse server 9 以及 windows NT 4.0、windows 2000、windows xp、windows 2003 环境下测试通过。

挂载UNIX系统NFS文件共享

类似于 windows 的网络共享，UNIX(Linux)系统也有自己的网络共享，那就是 NFS(网络文件系统)，下面我们就以 SUN Solaris2.8 和 REDHAT as server 3 为例简单介绍一下在 linux 下如何 mount nfs 网络共享。

在 linux 客户端挂载(mount)NFS 磁盘共享之前，必须先配置好 NFS 服务端。

1、Solaris 系统 NFS 服务端配置方法如下：

(1)修改 /etc/dfs/dfstab，增加共享目录

share -F nfs -o rw /export/home/sunky

(2)启动 nfs 服务

/etc/init.d/nfs.server start

(3)NFS 服务启动以后,也可以使用下面的命令增加新的共享

```
# share /export/home/sunky1
```

```
# share /export/home/sunky2
```

注: /export/home/sunky 和/export/home/sunky1 是准备共享的目录

2、linux 系统 NFS 服务端配置方法如下:

(1)修改 /etc/exports,增加共享目录

```
/export/home/sunky 10.140.133.23(rw)
```

```
/export/home/sunky1 *(rw)
```

```
/export/home/sunky2 linux-client(rw)
```

注: /export/home/目录下的 sunky、sunky1、sunky2 是准备共享的目录, 10.140.133.23、*、 linux-client 是被允许挂接此共享 linux 客户机的 IP 地址或主机名。如果要使用主机名 linux-client 必须在服务端主机 /etc/hosts 文件里增加 linux-client 主机 ip 定义。格式如下:

```
10.140.133.23 linux-client
```

(2)启动与停止 NFS 服务

```
/etc/rc.d/init.d/portmap start (在 REDHAT 中 PORTMAP 是默认启动的)
```

```
/etc/rc.d/init.d/nfs start 启动 NFS 服务
```

```
/etc/rc.d/init.d/nfs stop 停止 NFS 服务
```

注: 若修改/etc/export 文件增加新的共享, 应先停止 NFS 服务, 再启动 NFS 服务方能使新增加的共享起作用。使用命令 exportfs -rv 也可以达到同样的效果。

3、linux 客户端挂接(mount)其他 linux 系统或 UNIX 系统的 NFS 共享

```
# mkdir -p /mnt/nfs
```

注: 建立一个目录用来作挂接点(mount point)

```
#mount -t nfs -o rw 10.140.133.9:/export/home/sunky /mnt/nfs
```

注: 这里我们假设 10.140.133.9 是 NFS 服务端的主机 IP 地址, 当然这里也可以使用主机名, 但必须在本机/etc/hosts 文件里增加服务端 ip 定义。/export/home/sunky 为服务端共享的目录。

如此就可以在 linux 客户端通过/mnt/nfs 来访问其它 linux 系统或 UNIX 系统以 NFS 方式共享出来的文件了。以上操作在 redhat as server 3、redflag server4.1、suse server 9 以及 Solaris 7、Solaris 8、Solaris 9 for x86&sparc 环境下测试通过。

权限问题:

假设 server 端的使用者 jack, user id 为 1818, gid 为 1818, client 端也有一个使用者 jack, 但是 uid 及 gid 是 1818。client 端的 jack 希望能完全读写 server 端的 /home/jack 这个目录。server 端的 /etc/exports 是

这样写的:

```
/home/jack *(rw,all_squash,anonuid=1818,anongid=1818)
```

这个的配置文件的意思是, 所有 client 端的使用者存取 server 端 /home/jack 这目录时, 都会 map 成 server 端的 jack (uid,gid=1818)。我 mount 的结果是

1. client 端的 root 可以完全存取该目录, 包括读、写、杀……等

2. client 端的 jack (uid,gid=1818) 我可以做:

```
rm -rf server_jack/*
```

```
cp something server_jack/
```

```
mkdir server_jack/a
```

umount: 将文件设备卸载

```
[root@centos57 ~]# umount /dev/hda1 用设备文件名来卸载
```

```
[root@centos57 ~]# umount /aixi 用挂载点来卸载
```

umount: /aixi: device is busy 如果提示设备忙，不急可以使用如下命令卸载
#umount -l /mnt/hda1 选项 -l 并不是马上 umount，而是在该目录空闲后再 umount。
如果比较急，可用如下命令：
#umount -f /mnt/hda1 -f 代表强制卸载
如果还不行，可使用 fuser -m -v /dev/hda1 来查询是哪些程序在占用，结束这些程序进程即可卸载
[root@centos57 aixi]# sync && fuser -m /dev/hda1 -k 使用这条命令后一定可以卸载

交换分区

交换分区最大容量为 64G，最多只能建 32 个，

创建交换分区

#fdisk /dev/hda → n → +容量 → p → t(修改系统 ID) → 分区号 → 82 → p → w

#mkswap /dev/hda2 (以上划分的分区号) 构建 swap 格式

#swapon /dev/hda2 加载即完成增加 swap

#swapon -s 显示目前使用的 Swap 设备

创建交换文件

dd if=/dev/hda1 of=/aixi/swap bs=1M count=64 创建大文件

#mkswap /aixi/swap

#swapon /aixi/swap 完成

取消交换分区或者交换文件

#swapon -s 显示目前使用的 Swap 设备

#swapoff /aixi/swap

#swapoff /dev/hda2

#free -m 查看

三、用户管理

用户和用户组操作命令

Id

Finger

Pwck

检查/etc/passwd 配置文件内的信息与实际主文件夹是否存在，还可比较/etc/passwd 和/etc/shadow 的信息是否一致，另外如果/etc/passwd 中的数据字段错误也会提示。

Grpck

和 pwck 功能相近，这是检查用户组的。

Groups

newgrp

useradd

usermod

usermod 不仅能改用户的 **SHELL** 类型，所归属的用户组，也能改用户密码的有效期，还能改登录名。**usermod** 如此看来就是能做到用户帐号大转移；比如我把用户 **A** 改为新用户 **B**；

```
usermod [-u uid [-o]] [-g group] [-G group,...]
        [-d 主目录 [-m]] [-s shell] [-c 注释] [-l 新名称]
        [-f 失效日] [-e 过期日] [-p 密码] [-L|-U] 用户名
```

usermod 命令会参照你命令列上指定的部份修改系统帐号档。下列为 **usermod** 可选用的参数。

-c comment

更新用户帐号 password 档中的注解栏，一般是使用 **chfn(1)** 来修改。

-d home_dir

更新用户新的登入目录。如果给定 **-m** 选项，用户旧目录会搬到新的目录去，如旧目录不存在则建个新的。

-e expire_date 加上用户帐号停止日期。日期格式为 MM/DD/YY。

-f inactive_days 帐号过期几日后永久停权。当值为 0 时帐号则立刻被停权。而当值为 -1 时则关闭此功能。预设值为 -1。

-g initial_group 更新用户新的起始登入用户组。用户组名须已存在。用户组 ID 必须参照既有的用户组。用户组 ID 预设值为 1。

-G group, [...] 定义用户为一堆 groups 的成员。每个用户组使用“,”区格开来，不可以夹杂空白字元。用户组名同 **-g** 选项的限制。如果用户现在的用户组不再此列，则将用户由该用户组中移除。

-l login_name 变更用户 login 时的名称为 login_name。其它不变。特别是，用户目录名应该也会跟着更动成新的登入名。

-s shell 指定新登入 shell。如此栏留白，系统将选用系统预设 shell。

-u uid 用户 ID 值。必须为唯一的 ID 值，除非用 **-o** 选项。数字不可为负值。预设值为最小不得小于 **/etc/login.defs** 中定义的 **UID_MIN** 值。0 到 **UID_MIN** 值之间是传统上保留给系统帐号使用。用户目录树下所有的档案目录其 **userID** 会自动改变。放在用户目录外的档案则要自行手动更动。

警告：**usermod** 不允许你改变正在线上的用户帐号名称。当 **usermod** 用来改变 **userID**，必须确认这名 **user** 没在电脑上执行任何程序。你需手动更改用户的 **crontab** 档。也需手动更改用户的 **at** 工作档。采用 **NISserver** 须在 **server** 上更动相关的 **NIS** 设定。

举个简单的例子，我们在前面说了关于 **useradd** 的工具，而 **usermod** 工具和 **useradd** 的参数差不多；两者不同之处在于 **useradd** 是添加，**usermod** 是修改；

```
[root@localhost ~]# usermod -d /opt/linuxfish -m -l fishlinux -U linuxfish
```

注：把 linuxfish 用户名改为 fishlinux，并且把其家目录转移到 /opt/linuxfish；

```
[root@localhost ~]# ls -la /opt/linuxfish/ 注：查看用户 fishlinux 的家目录下的文件及属主；
```

总用量 48

```
drwxr-xr-x  3 fishlinux linuxfish 4096 11月  5 16:46 .
drwxrwxrwx 29 root      root      4096 11月  5 16:48 ..
-rw-r--r--  1 fishlinux linuxfish  24 11月  5 16:46 .bash_logout
-rw-r--r--  1 fishlinux linuxfish 191 11月  5 16:46 .bash_profile
-rw-r--r--  1 fishlinux linuxfish 124 11月  5 16:46 .bashrc
-rw-r--r--  1 fishlinux linuxfish 5619 11月  5 16:46 .canna
-rw-r--r--  1 fishlinux linuxfish  438 11月  5 16:46 .emacs
```

```
-rw-r--r--  1 fishlinux linuxfish  120 11月  5 16:46 .gtkrc
drwxr-xr-x  3 fishlinux linuxfish 4096 11月  5 16:46 .kde
-rw-r--r--  1 fishlinux linuxfish   0 11月  5 16:46 mydoc.txt
-rw-r--r--  1 fishlinux linuxfish  658 11月  5 16:46 .zshrc
[root@localhost ~]# more /etc/passwd |grep fishlinux 注：查看有关 fishlinux 的记录；
fishlinux:x:512:512::/opt/linuxfish:/bin/bash
```

通过上面的例子，我们发现文件的用户组还没有变，如果您想改变为 **fishlinux** 用户组，如果想用通过 **usermod** 来修改，就要先添加 **fishlinux** 用户组；然后用 **usermod -g** 来修改，也可以用 **chown -R fishlinux:fishlinux /opt/finshlinux** 来改；

警告： **usermod** 最好不要用它来改用户的密码，因为他在 **/etc/shadow** 中显示的是明口令；修改用户的口令最好用 **passwd**；

```
[root@localhost ~]# usermod -p 123456 fishlinux 注：修改 fishlinux 的口令是 123456；
[root@localhost ~]# more /etc/shadow |grep fishlinux 注：查询/etc/shadow 文件中 fishlinux 的口令；我们看到明显
是没有加密；
fishlinux:123456:13092:0:99999:7:::
```

userdel

userdel 很简单，只有一个参数可选 **-r**；如果加参数 **-r**，表示在删除用户的同时，一并把用户的家目录及本地邮件存储的目录或文件也一同删除；比如我们现在有两个用户 **bnnb** 和 **lanhaitun**，其家目录都位于 **/home** 目录中，现在我们来删除这两个用户；

```
[root@localhost ~]# userdel bnnb 注：删除用户 bnnb，但不删除其家目录及文件；
[root@localhost ~]# ls -ld /home/bnnb 注：查看其家目录是否存在；
drwxr-xr-x 14 501 501 4096  8月 29 16:33 /home/bnnb 注：存在；
[root@localhost ~]# ls -ld /home/lanhaitun 注：查看 lanhaitun 家目录是否存在；
drwx----- 4 lanhaitun lanhaitun 4096 11月  5 14:50 /home/lanhaitun 注：存在；
[root@localhost ~]# userdel -r lanhaitun 注：删除用户 lanhaitun，其家目录及文件一并删除；
[root@localhost ~]# ls -ld /home/lanhaitun 注：查看是否在删除 lanhaitun 用户的同时，也一并把其家目录和文件一
同删除；
ls: /home/lanhaitun: 没有那个文件或目录 注：已经删除；
```

警告： 请不要轻易用 **-r** 参数；他会删除用户的同时删除用户所有的文件和目录，切记；如果用户目录下有重要的文件，在删除前请备份；

其实也有最简单的办法，但这种办法有点不安全，也就是直接在 **/etc/passwd** 中删除您想要删除用户的记录；但最好不要这样做，**/etc/passwd** 是极为重要的文件，可能您一不小心会操作失误；

Groupadd

groupmod

groupdel 是用来删除用户组的；

语法格式： **groupdel** 用户组

比如：

```
[root@localhost ~]# groupdel lanhaitun
```

passwd

passwd 作为普通用户和超级权限用户都可以运行，但作为普通用户只能更改自己的用户密码，但前提是没有被 **root** 用户锁定；如果 **root** 用户运行 **passwd**，可以设置或修改任何用户的密码；

passwd 命令后面不接任何参数或用户名，则表示修改当前用户的密码；请看下面的例子；

```
[root@localhost ~]# passwd 注：没有加任何用户，我是用 root 用户来执行的 passwd 表示修改 root 用户的密码；下面也有提示；
```

```
Changing password for user root.
```

```
New UNIX password: 注：请输入新密码；
```

```
Retype new UNIX password: 注：验证新密码；
```

```
passwd: all authentication tokens updated successfully. 注：修改 root 密码成功；
```

如果是普通用户执行 **passwd** 只能修改自己的密码；

如果新建用户后，要为新用户创建密码，则用 **passwd 用户名**，注意要以 **root** 用户的权限来创建；

```
[root@localhost ~]# passwd beinan 注：更改或创建 beinan 用户的密码；
```

```
Changing password for user beinan.
```

```
New UNIX password: 注：请输入新密码；
```

```
Retype new UNIX password: 注：再输入一次；
```

```
passwd: all authentication tokens updated successfully. 注：成功；
```

普通用户如果想更改自己的密码，直接运行 **passwd** 即可；比如当前操作的用户是 **beinan**；

```
[beinan@localhost ~]$ passwd
```

```
Changing password for user beinan. 注：更改 beinan 用户的密码；
```

```
(current) UNIX password: 注：请输入当前密码；
```

```
New UNIX password: 注：请输入新密码；
```

```
Retype new UNIX password: 注：确认新密码；
```

```
passwd: all authentication tokens updated successfully. 注：更改成功；
```

passwd 几个比较重要的参数；

```
[root@localhost beinan]# passwd --help
```

```
Usage: passwd [OPTION...] <accountName>
```

-k, --keep-tokens keep non-expired authentication tokens

注：保留即将过期的用户在期满后仍能使用；

-d, --delete delete the password for the named account (root only)

注：删除用户密码，仅能以 root 权限操作；

-l, --lock lock the named account (root only)

注：锁住用户无权更改其密码，仅能通过 root 权限操作；

-u, --unlock unlock the named account (root only)

注：解除锁定；

-f, --force force operation

注：强制操作；仅 root 权限才能操作；

-x, --maximum=DAYS maximum password lifetime (root only) 注：两次密码修正的最大天数，后面接数字；仅能 root 权限操作；

-n, --minimum=DAYS minimum password lifetime (root only) 注：两次密码修改的最小天数，后面接数字；仅能 root 权限操作；

<code>-w, --warning=DAYS</code>	number of days warning users receives before	注：在距多少天提醒用户修改密码；仅能 root 权限操作；
	password expiration (root only)	
<code>-i, --inactive=DAYS</code>	number of days after password expiration when an	注：在密码过期后多少天，用户被禁掉，仅能以 root 操作；
	account becomes disabled (root only)	
<code>-S, --status</code>	report password status on the named account (root	注：查询用户的密码状态，仅能 root 用户操作；
	only)	
<code>--stdin</code>	read new tokens from stdin (root only)	

比如我们让某个用户不能修改密码，可以用 `-l` 参数来锁定：

```
[root@localhost ~]# passwd -l beinan 注：锁定用户 beinan 不能更改密码；
Locking password for user beinan.
passwd: Success 注：锁定成功；
[beinan@localhost ~]# su beinan 注：通过 su 切换到 beinan 用户；
[beinan@localhost ~]$ passwd 注：beinan 来更改密码；
Changing password for user beinan.
Changing password for beinan
(current) UNIX password: 注：输入 beinan 的当前密码；
passwd: Authentication token manipulation error 注：失败，不能更改密码；
```

再来一例：

```
[root@localhost ~]# passwd -d beinan 注：清除 beinan 用户密码；
Removing password for user beinan.
passwd: Success 注：清除成功；
[root@localhost ~]# passwd -S beinan 注：查询 beinan 用户密码状态；
Empty password. 注：空密码，也就是没有密码；
```

注意： 当我们清除一个用户的密码时，登录时就无需密码；这一点要加以注意；

chage 修改用户密码有效期限的命令；

chage 用语法格式：

```
chage [-l] [-m 最小天数] [-M 最大天数] [-W 警告] [-I 失效日] [-E 过期日] [-d 最后日] 用户
```

前面已经说的好多了，这个只是一笔带过吧，知道有这个命令就行，自己实践实践再说，大体和 `passwd` 有些参数的用法差不多；

id 工具：查询用户所对应的UID 和GID 及GID所对应的用户组；

`id` 工具是用来查询用户信息，比如用户所归属的用户组，`UID` 和 `GID` 等；`id` 用法极为简单；我们举个例子说明一下；

语法格式： `id [参数] [用户名]`

至于有哪些参数，自己查一下 `id --help` 或 `man id`；如果 `id` 后面不接任何参数和任何用户，默认显示当前操作用户的用户名、所归属的用户组、`UID` 和 `GID` 等；

实例一：不加任何参数和用户名；

```
[beinan@localhost ~]$ id
```

```
uid=500(beinan) gid=500(beinan) groups=500(beinan)
```

注解：在没有加任何参数的情况下，查询的是当前操作用户的用户名、UID、GID 和所处的主用户组和附属用户组；在本例中，用户名是 beinan，UID 是 500，所归属的主用户组是 beinan，GID 是 500；

实例二：id 后面接用户名：

如果我们想查询系统中用户的 UID 和 GID 相应的内容，可以直接接用户名，但用户名必须是真实的，能在 /etc/passwd 中查到的；

```
[beinan@localhost ~]$ id linuxsir
```

```
uid=505(linuxsir) gid=502(linuxsir) groups=502(linuxsir),0(root),500(beinan)
```

注解：查询用户 linuxsir 的信息，用户 linuxsir，UID 为 505，所归属的主用户组是 linuxsir，主用户组的 GID 是 502；同时 linuxsir 用户也是 GID 为 0 的 root 用户组成员，也是 GID 为 500 用户组 beinan 的成员；

这个例子和实例一在用户组方面有所不同，我们在 [《Linux 用户（user）和用户组（group）管理概述》](#) 中有提到；用户和用户组的对应关系，可以是一对一、一对多、多对一、或多对多的交叉关系，请参考之；另外您还需要掌握 [《用户（user）和用户组（group）配置文件详解》](#) 一文；

2、finger 工具：用来查询用户信息，侧重用户家目录、登录 SHELL 等；

finger 工具侧重于用户信息的查询；查询的内容包括用户名（也被称为登录名 Login），家目录，用户真实的名字（Name）... 办公地址、办公电话；也包括登录终端、写状态、空闲时间等；

我们最常用 finger 来查询用户家目录、用户真实名、所用 SHELL 类型、以及办公地址和电话，这是以参数 -l 长格式输出的；而修改用户的家目录、真实名字、办公地址及办公电话，我们一般要能通过 chfn 命令进行；

语法格式：

finger [参数选项] [用户名]

-l 采用长格式（默认），显示由 -s 选项所包含的所有信息，以及主目录、办公地址、办公电话、登录 SHELL、邮件状态、.plan、.project 和 .forward；

-m 禁止对用户真实名字进行匹配；

-p 把 .plan 和 .project 文件中的内容省略；

-s 显示短格式，用户名（也被称为登录名 Login）、真实名字（NAME）、在哪个终端登录（Tty）、写状态、空闲时间（Idle）、登录时间（Login Time）、办公地点、办公电话等；

至于 finger 有哪些参数，您可以通过 finger --help 或 man finger 来获取，我们在本文中以实例讲述最常用的参数；

实例一：不接任何参数，也不指定查询用户名；默认为加了 -s 参数；

```
[beinan@localhost ~]$ finger
```

Login	Name	Tty	Idle	Login Time	Office	Office Phone
beinan	beinan sun	tty1	1:39	Nov 2 08:27		
linuxsir	linuxsir open	tty2	2	Nov 2 10:03	linuxsir o	+1-389-866-771

等价命令

```
[beinan@localhost ~]$ finger -s
```

注解：不加任何参数，也没有指定查询哪个用户，finger 会以默认以短格 -s 来输出登录本机的所有用户的用户名（也被称为登录名 Login）、真实名字（NAME）、在哪个终端登录（Tty）、写状态、空闲时间（Idle）、登录时间（Login Time）、办公地点、办公电话等；

在这个例子中，有 beinan 用户登录，真实名字是 beinan sun（这个名字是用户的真实名字，如果在添加用户时没有设置，是不会显示的），在 tty1 终端登录，空闲时间是 1 分 39 秒，登录时间是 Nov /2/08:27，没有办公室名称，没有办公电话；请对照本例中 beinan 用户记录的解说，我们来看看本例中的 linuxsir 用户信息；应该不难。

关于写状态，如果在 Tty 后面没有任何输出，表示正在写入，如果有 * 出现，表示没有写入或被禁止，比如下面的例子，ftp 用户没有通过终端登录系统，因为 Tty 是 *，同时 Tty 后面还有一个 *，表示禁止写入或没有写入状态（当用户没有登录时）；

```
[beinan@localhost ~]$ finger -s ftp
```

Login	Name	Tty	Idle	Login Time	Office	Office Phone
ftp	FTP User	*	* No logins			

我们可以以短格式的来查询某个用户信息以短格式输出，比如下面的例子；

```
[beinan@localhost ~]$ finger -s beinan
```

实例二：关于长格式的用户信息的输出 **-l** 参数的实例；

finger -l 如果不加用户名的情况下，可以列出所有通过 **tty** 登录的用户信息；如果您想查询某个用户，就直接指定用户，可以指定一个或多个；什么是 **tty** 登录？如果您在全屏文本界面操作的话，您可以通过按 **CTRL+F2** 或 **CTRL+F3** 或 **CTRL+F4** 等，以几个不同的用户登录到主机上，您就会看到，每个用户都有不同的 **tty**；

```
[beinan@localhost ~]$ finger -l
[beinan@localhost ~]$ finger -l beinan linuxsir 注：可以同时查询几个用户信息，以长格式输出；
[beinan@localhost ~]$ finger beinan
```

```
Login: beinan                               Name: beinan sun
Directory: /home/beinan                     Shell: /bin/bash
On since Wed Nov  2 08:27 (CST) on tty1      2 hours 29 minutes idle
On since Wed Nov  2 10:50 (CST) on pts/0 from :0.0
No mail.
No Plan.
```

在本例中，所查询的用户是 **beinan**，真实名字是 **beinan sun**，家目录位于 **/home/beinan**，所用 **SHELL** 类型是 **bash**；然后就是通过哪个终端登录的，登录时间，是不是有 **mail**，有 **Plan** 等；

实例三：参数组合的例子；

```
[beinan@localhost ~]$ finger -lp beinan
Login: beinan                               Name: beinan sun
Directory: /home/beinan                     Shell: /bin/bash
On since Wed Nov  2 08:27 (CST) on tty1      2 hours 36 minutes idle
On since Wed Nov  2 10:50 (CST) on pts/0 from :0.0
No mail.
```

注解：查询 **beinan** 用户信息，以长格式输出，并且不输出 **Plan** 和 **Project** 的内容；

实例四：**finger -s** 和 **w** 及 **who** 的比较；

对于 **finger** 就说这么多吧，极为简单的工具，当用到 **-s** 参数时，您最好和 **w** 和 **who** 工具对照，看看 **finger -s** 和 **w** 及 **who** 的输出有什么异同，**w** 和 **who** 是查询哪些用户登录主机的；而 **finger -s** 呢，无论是登录还是不登录的用户都可以查；但所查到的内容侧重有所不同；自己看看例子；

```
[beinan@localhost ~]$ finger -s
Login      Name           Tty      Idle  Login Time  Office      Office Phone
beinan     beinan sun     tty1     3:03  Nov  2 08:27
beinan     beinan sun     pts/0    Nov  2 10:50 (:0.0)
linuxsir   linuxsir open  tty2     1:26  Nov  2 10:03 linuxsir o +1-389-866-771
[beinan@localhost ~]$ w
11:30:36 up  3:04,  3 users,  load average: 0.30, 0.15, 0.10
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
beinan    tty1     -             08:27    3:03m  2:52   0.00s /bin/sh /usr/X11R6/bin/startx
linuxsir  tty2     -             10:03    1:26m  0.01s  0.01s -bash
beinan    pts/0    :0.0          10:50    0.00s  0.16s  0.00s w
[beinan@localhost ~]$ who
beinan    tty1     Nov  2 08:27
```



```
linuxsir tty2          Nov  2 10:03
beinan pts/0           Nov  2 10:50 (:0.0)
```

3、查询登录主机的用户工具：w、who、users

w、who 和 users 工具，是查询已登录当前主机的用户；另外 finger -s 也同样能查询；侧重点不一样；请自己对比着看；毕竟简单，这里只是介绍；

```
[beinan@localhost ~]$ w
12:09:56 up 3:43, 7 users, load average: 0.16, 0.10, 0.04
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
beinan    tty1     -               08:27    3:42m  3:09   0.00s /bin/sh /usr/X11R6/bin/startx
linuxsir  tty2     -               10:03    2:06m  0.01s  0.01s -bash
beinan    pts/0    :0.0            11:36    1:09   0.15s  0.15s bash
beinan    pts/1    :0.0            11:37    1:12   0.21s  0.21s bash
beinan    pts/2    :0.0            12:02    6:52   0.09s  0.09s bash
beinan    pts/3    :0.0            12:05    12.00s  0.11s  0.06s ssh xmbnnbdl@linuxsir.org -p 17007
beinan    pts/4    :0.0            12:06    0.00s  0.21s  0.00s w

[beinan@localhost ~]$ who
beinan    tty1          Nov  2 08:27
linuxsir  tty2          Nov  2 10:03
beinan    pts/0         Nov  2 11:36 (:0.0)
beinan    pts/1         Nov  2 11:37 (:0.0)
beinan    pts/2         Nov  2 12:02 (:0.0)
beinan    pts/3         Nov  2 12:05 (:0.0)
beinan    pts/4         Nov  2 12:06 (:0.0)

[beinan@localhost ~]$ users
beinan beinan beinan beinan beinan beinan linuxsir
```

4、groups 用户所归属的用户组查询；

groups 用法很简单，就是查询用户所归属哪个或哪些用户组；

语法格式： groups 用户名

实例：

```
[beinan@localhost ~]$ groups beinan 注：查询 beinan 所归属的用户组；
beinan : beinan 注：beinan 是 beinan 用户组下的成员；

[beinan@localhost ~]$ groups linuxsir 注：查询 linuxsir 用户所归属的用户组；
linuxsir : linuxsir root beinan 注：linuxsir 用户是 linuxsir 用户组、beinan 用户组、root 用户组成员；
```

groups 主要是查询用户所归属的用户组名，最好和 id 命令相对比；这样对这两个工具都有所了解

相关配置文件

```
/etc/passwd
/etc/shadow
/etc/gshadow
/etc/group
```

Linux 用户密码策略

Linux 用户密码的有效期限,是否可以修改密码可以通过 login.defs 文件控制.对 login.defs 文件修只影响后续建立的用户,如果要改变以前建立的用户的有效期限等可以使用 chage 命令.

Linux 用户密码的复杂度可以通过 `pam pam_cracklib module` 或 `pam_passwdqc module` 控制,两者不能同时使用. 个人感觉 `pam_passwdqc` 更好用.

`/etc/login.defs` 密码策略

```
PASS_MAX_DAYS 99999    #密码的最大有效期, 99999:永久有期
PASS_MIN_DAYS  0       #是否可修改密码,0 可修改,非 0 多少天后可修改
PASS_MIN_LEN   5       #密码最小长度,使用 pam_cracklib module,该参数不再有效
PASS_WARN_AGE  7       #密码失效前多少天在用户登录时通知用户修改密码
```

`pam_cracklib` 主要参数说明:

```
retry=N:重试多少次后返回密码修改错误
difok=N:新密码必需与旧密码不同的位数
dcredit=N: N >= 0:密码中最多有多少个数字;N < 0 密码中最少有多少个数字.
lcredit=N:小写字母的个数
ucredit=N 大写字母的个数
credit=N:特殊字母的个数
minclass=N:密码组成(大/小字母,数字,特殊字符)
```

`pam_passwdqc` 主要参数说明:

```
mix:设置口令字最小长度, 默认值是 mix=disabled。
max:设置口令字的最大长度, 默认值是 max=40。
passphrase:设置口令短语中单词的最少个数, 默认值是 passphrase=3, 如果为 0 则禁用口令短语。
atch:设置密码串的常见程序, 默认值是 match=4。
similar:设置当我们重设口令时, 重新设置的新口令能否与旧口令相似, 它可以是 similar=permit 允许相似或 similar=deny 不允许相似。
random:设置随机生成口令字的默认长度。默认值是 random=42。设为 0 则禁止该功能。
enforce:设置约束范围, enforce=none 表示只警告弱口令字, 但不禁止它们使用; enforce=users 将对系统上的全体非根用户实行这一限制; enforce=everyone 将对包括根用户在内的全体用户实行这一限制。
non-unix:它告诉这个模块不要使用传统的 getpwnam 函数调用获得用户信息,
retry:设置用户输入口令字时允许重试的次数, 默认值是 retry=3
```

密码复杂度通过`/etc/pam.d/system-auth` 实施

如:

要使用 `pam_cracklib` 将注释去掉,把 `pam_passwdqc.so` 注释掉即可.

```
#password requisite /lib/security/$ISA/pam_cracklib.so retry=3 difok=1
password requisite /lib/security/$ISA/pam_passwdqc.so min=disabled,24,12,8,7 passphrase=3
password sufficient /lib/security/$ISA/pam_unix.so nullok use_authtok md5 shadow
```

```
#password requisite /lib/security/$ISA/pam_cracklib.so retry=3 difok=1
```

新密码至少有一位与原来的不同

`PASS_MIN_DAYS` 参数则设定了在本次密码修改后, 下次允许更改密码之前所需的最少天数。

`PASS_WARN_AGE` 的设定则指明了在口令失效前多少天开始通知用户更改密码 (一般在用户刚刚登陆系统时就会收到警告通知)。

你也会编辑`/etc/default/useradd` 文件, 寻找 `INACTIVE` 和 `EXPIRE` 两个关键词:

`INACTIVE=14`

`EXPIRE=`

这会指明在口令失效后多久时间内，如果口令没有进行更改，则将账户更改为失效状态。在本例中，这个时间是 14 天。而 EXPIRE 的设置则用于为所有新用户设定一个密码失效的明确时间（具体格式为“年份-月份-日期”）。

显然，上述这些设定更改之后，只能影响到新建的用户。要想修改目前已存在的用户具体设置，需要使用 chage 工具。

```
# chage -M 60 joe
```

这条命令将设置用户 joe 的 PASS_MAX_DAYS 为 60，并修改对应的 shadow 文件。

你可以使用 chage -l 的选项，列出当前的账户时效情况，而使用 -m 选项是设置 PASS_MIN_DAYS，用 -w 则是设置 PASS_WARN_AGE，等等。chage 工具可以让你修改特定账户的所有密码时效状态。

注意，chage 仅仅适用于本地系统的账户，如果你在使用一个类似 LDAP 这样的认证系统时，该工具会失效。如果你在使用 LDAP 作为认证，而你打算使用 chage，那么，哪怕仅仅是试图列出用户密码的时效信息，你也会发现 chage 根本不起作用。

制定一项策略，定义多长时间一个密码必须进行更改，然后强制执行该策略，是非常不错的一个做法。在解雇了某个雇员后，口令时效策略会保证该雇员不可能在被解雇 3 个月后发现他的口令依然可用。即使系统管理员忽略了删除他的帐号，该帐号也会因密码时效策略而被自动锁定。当然，这一点并不能成为不及时删除该雇员帐号的理由，但是这个策略的确提供了一层额外的安全防护，尤其是在过去常常忽视及时清理帐号的情况下。

ACL权限设置

ACL 是 Access Control List 的缩写，主要用于在提供传统的 owner、group、others 的 read、write、execute 权限之外进行细部权限设置。

启动ACL

让/目录支持 ACL:

```
#mount -o remount ,acl /
```

```
#mount |grep / //查看是否有挂载
```

开机启动 ACL:

将要启动 ACL 的分区写入/etc/fstab 中:

```
#vi /etc/fstab
```

```
/dev/hda5      /              ext3          default,acl   1            2
```

ACL相关命令

Getfacl:取得某个文件/目录的 ACL 权限;

Setfacl:设置某个文件/目录的 ACL 权限;

```
setfacl [-bkndRLPvh] [{-m|-x} acl_spec] [{-M|-X} acl_file] file ...
```

```
setfacl --restore=file
```

描述

setfacl 用来在命令行里设置 ACL。在命令行里，一系列的命令跟随以一系列的文件名。

选项 -m 和 -x 后边跟以 acl 规则。多条 acl 规则以逗号(,)隔开。选项 -M 和 -X 用来从文件或标准输入读取 acl 规则。

选项 --set 和 --set-file 用来设置文件或目录的 acl 规则，先前的设定将被覆盖。

选项 -m(--modify)和 -M(--modify-file)选项修改文件或目录的 acl 规则。

选项 -x(--remove)和 -X(--remove-file)选项删除 acl 规则。

当使用 -M, -X 选项从文件中读取规则时，setfacl 接受 getfacl 命令输出的格式。每行至少一条规则，以# 开始的行将被视为注释。

当在不支持 ACLs 的文件系统上使用 **setfacl** 命令时，**setfacl** 将修改文件权限位。如果 **acl** 规则并不完全匹配文件权限位，**setfacl** 将会修改文件权限位使其尽可能的反应 **acl** 规则，并向 **standard error** 发送错误消息，以大于 0 的状态返回。

权限

文件的所有者以及有 **CAP_FOWNER** 的用户进程可以设置一个文件的 **acl**。（在目前的 linux 系统上，root 用户是唯一有 **CAP_FOWNER** 能力的用户）

选项

-b, --remove-all

删除所有扩展的 **acl** 规则，基本的 **acl** 规则(所有者，群组，其他)将被保留。

-k, --remove-default

删除缺省的 **acl** 规则。如果没有缺省规则，将不提示。

-n, --no-mask

不要重新计算有效权限。**setfacl** 默认会重新计算 **ACL mask**，除非 **mask** 被明确的制定。

--mask

重新计算有效权限，即使 **ACL mask** 被明确指定。

-d, --default

设定默认的 **acl** 规则。

--restore=file

从文件恢复备份的 **acl** 规则（这些文件可由 **getfacl -R** 产生）。通过这种机制可以恢复整个目录树的 **acl** 规则。此参数不能和除 **--test** 以外的任何参数一同执行。

--test

测试模式，不会改变任何文件的 **acl** 规则，操作后的 **acl** 规格将被列出。

-R, --recursive

递归的对所有文件及目录进行操作。

-L, --logical

跟踪符号链接，默认情况下只跟踪符号链接文件，跳过符号链接目录。

-P, --physical

跳过所有符号链接，包括符号链接文件。

--version

输出 **setfacl** 的版本号并退出。

--help

输出帮助信息。

--

标识命令行参数结束，其后的所有参数都将被认为是文件名

-

如果文件名是 **-**，则 **setfacl** 将从标准输入读取文件名。

ACL 规则

setfacl 命令可以识别以下的规则格式。

[d[efault]:] [u[ser]:]uid [:perms]

指定用户的权限，文件所有者的权限（如果 **uid** 没有指定）。

[d[efault]:] g[r[ou]p:]gid [:perms]

指定群组的权限，文件所有群组的权限（如果 **gid** 未指定）

[d[efault]:] m[ask][:] [:perms]

有效权限掩码

[d[efault]:] o[ther] [:perms]

其他的权限

恰当的 **acl** 规则被用在修改和设定的操作中。

对于 **uid** 和 **gid**，可以指定一个数字，也可指定一个名字。

perms 域是一个代表各种权限的字母的组合：读-**r** 写-**w** 执行-**x**，执行只适合目录和一些可执行的文件。

pers 域也可设置为八进制格式。

自动创建的规则

最初的，文件目录仅包含 3 个基本的 **acl** 规则。为了使规则能正常执行，需要满足以下规则。

*3 个基本规则不能被删除。

*任何一条包含指定的用户名或群组名的规则必须包含有效的权限组合。

*任何一条包含缺省规则的规则在使用时，缺省规则必须存在。

用法举例如下：

acl 全称 Access Control Lists 翻译成中文叫"访问控制列表"，

传统的 Linux 文件系统的权限控制是通过 **user**、**group**、**other** 与 **r**(读)、**w**(写)、**x**(执行) 的不同组合来实现的。随着应用的发展，这些权限组合已不能适应现时复杂的文件系统权限控制要求。例如，目录 **/data** 的权限为：**drwxr-x---**，所有者与所属组均为 **root**，在不改变所有者的前提下，要求用户 **tom** 对该目录有完全访问权限 (**rw****x**)。考虑以下 2 种办法 (这里假设 **tom** 不属于 **root group**)

(1) 给 **/data** 的 **other** 类别增加 **rw****x** permission，这样由于 **tom** 会被归为 **other** 类别，那么他也将拥有 **rw****x** 权限。

(2) 将 **tom** 加入到 **root group**，为 **root group** 分配 **rw****x** 权限，那么他也将拥有 **rw****x** 权限。

以上 2 种方法其实都不合适，所以传统的权限管理设置起来就力不从心了。

为了解决这些问题，Linux 开发出了一套新的文件系统权限管理方法，叫文件访问控制列表 (Access Control Lists, **ACL**)。简单地来说，**ACL** 就是可以设置特定用户或者用户组对于一个文件的操作权限。

ACL 有两种，一种是存取 **ACL** (access **ACLs**)，针对文件和目录设置访问控制列表。一种是默认 **ACL** (default **ACLs**)，只能针对目录设置。如果目录中的文件没有设置 **ACL**，它就会使用该目录的默认 **ACL**。

首先我来讲一下 **getfacl** (显示文件或目录的 **ACL**)

在我的电脑里首先有一个用户叫 **NEU**。我们学校的简称。同时还有一个用户，**software**，我的专业名称。

我以 **neu** 用户进行操作，在其目录下建立一个文件 **fileofneu**。

可以看到它的初始权限为 **-rw-rw-r--**。然后我把这个文件权限进行下修改。使用的命令为 **chmod**，修改后的文件权限为 **-rw-rw----**。现在这个文件的权限就不允许其它用户访问了。

然后切换到 **software** 用户，来证实这个文件的不可访问性。

下面我们就通过 **getfacl** 命令来查看。这时候得进入 **neu** 用户下操作了。其命令格式很简单：**getfacl fileofneu**

权限一目了然。不多介绍了，下面就要用 **Setfacl** 来进行修改了。使其在对于其它用户的权限里，只对 **software** 用户只读只写。

setfacl -m u:software:rw- fileofneu

setfacl -R -m u:software:rw- fileofneu (**-R** 一定要在 **-m** 前面，表示目录下所有文件)

setfacl -x u:software fileofneu (去掉单个权限)

setfacl -b (去掉所有 **acl** 权限)

如果我们希望在一个目录中新建的文件和目录都使用同一个预定的 **ACL**，那么我们可以使用默认(Default) **ACL**。在对一个目录设置了默认的 **ACL** 以后，每个在目录中创建的文件都会自动继承目录的默认 **ACL** 作为自己的 **ACL**。用 **setfacl** 的 **-d** 选项就可以做到这一点：

```
[root@FC3-vm mnt]# setfacl -d --set g:testg1:rw dir1
```

```
[root@FC3-vm mnt]# getfacl dir1
```

然后用 `getfacl` 命令来进行查看.我们就可以看到多了一行 `user:software:rw-` 这说明其对用户 `software` 开放了读写的权限.

为了证实其可用性,再切换到 `software` 用户下访问这个文件,发现与前面不同的是,这回可以读写了.

今天就讲这一个吧,讲多了,大家也记不住.

对了,刚才我进行了一下这个操作,发现进不去,原来是我没有给 `software` 用户授与访问 `/home/neu` 这个目录的进入的权力,所以,我们还得应用 `setfacl` 命令来使得 `software` 用户拥有进入这个目录的权力.其操作与上面基本一致.

用户身份切换

Su

命令作用

`su` 的作用是变更为其它使用者的身份,超级用户除外,需要键入该使用者的密码.

使用方式

`su [-fmp] [-c command] [-s shell] [--help] [--version] [-] [USER [ARG]]`

参数说明

`-f` , `-fast`: 不必读启动文件(如 `csch.cshrc` 等),仅用于 `csch` 或 `tcsh` 两种 Shell.

`-l` , `-login`: 加了这个参数之后,就好像是重新登陆一样,大部分环境变量(例如 `HOME`、`SHELL` 和 `USER` 等)都是以该使用者(`USER`)为主,并且工作目录也会改变.如果没有指定 `USER`,缺省情况是 `root`.

`-m`, `-p` , `-preserve-environment`: 执行 `su` 时不改变环境变数.

`-c command`: 变更账号为 `USER` 的使用者,并执行指令(`command`)后再变回原来使用者.

`-help` 显示说明文件

`-version` 显示版本资讯

`USER`: 欲变更的使用者账号,

`ARG`: 传入新的 Shell 参数.

例子

`su -c ls root` 变更帐号为 `root` 并在执行 `ls` 指令后退出变回原使用者.

`[user1@centos6 ~]$ su - root -c "head -n 3 /etc/passwd"` 对于命令参数要加上引号

`su [用户名]`

a>在 `root` 用户下,输入 `su` 普通用户.则切换至普通用户,从 `root` 切换到普通用户不需要密码

b>在普通用户下,输入 `su [用户名]`

提示 `password`:

输入用户的 `PASSWORD`,则切换至该用户

Sudo

`/etc/sudoers` 谁能作什么的一个列表, `Sudo` 能用需要在这个文件中定义

`#visudo` 增加如下,加%代表用户组, `ALL=(ALL)`表示登录者的来源主机名,最后的 `ALL` 代表可执行的命令. `NOPASSWD` 代表不需要密码直接可运行 `Sudo`,限制多命令一定要写绝对路径,用逗号分开,多行用 `'\'`,用 `!` 代表不能执行

`%aixi ALL=(ALL) NOPASSWD: ALL`

`%aixi ALL=(ALL) NOPASSWD: /bin/ls,/bin/mkdir,/bin/rmdir,\`

`/usr/bin/who,!/usr/bin/passwd root`

查询用户命令

W

可显示开机多久，当前登录的所有用户，平均负载

Who

显示当前登录的所有用户

Last

显示每个用户最后的登录时间

Lastlog

显示每个用户最后的登录时间

四、文件权限

1、文件类型

Linux 广泛的被很多用户所接受，它强大的功能受到很多人喜欢，Linux 文件一般是用一些相关的应用程序创建，比如图像工具、文档工具、归档工具... .. 或 cp 工具等。Linux 文件的删除方式是用 rm 命令。

Linux 文件类型和 Linux 文件的文件名所代表的意义是两个不同的概念。我们通过一般应用程序而创建的比如 file.txt、file.tar.gz，这些文件虽然要用不同的程序来打开，但放在 Linux 文件类型中衡量的话，大多是常规文件（也被称为普通文件）。

Linux 文件类型常见的有：普通文件、目录、字符设备文件、块设备文件、符号链接文件等；现在我们将进行一个简要的说明：

1 普通文件

1. [root@localhost ~]# ls -lh install.log
2. -rw-r--r-- 1 root root 53K 03-16 08:54 install.log

我们用 ls -lh 来查看某个文件的属性，可以看到有类似 -rw-r--r--，值得注意的是第一个符号是 -，这样的文件在 Linux 中就是普通文件。这些文件一般是用一些相关的应用程序创建，比如图像工具、文档工具、归档工具... .. 或 cp 工具等。这类文件的删除方式是用 rm 命令；

2 目录

1. [root@localhost ~]# ls -lh
2. 总计 14M
3. -rw-r--r-- 1 root root 2 03-27 02:00 fonts.scale
4. -rw-r--r-- 1 root root 53K 03-16 08:54 install.log
5. -rw-r--r-- 1 root root 14M 03-16 07:53 kernel-6.15-1.2025_FC5.i686.rpm
6. drwxr-xr-x 2 1000 users 4.0K 04-04 23:30 mkum1-2004.07.17
7. drwxr-xr-x 2 root root 4.0K 04-19 10:53 mydir
8. drwxr-xr-x 2 root root 4.0K 03-17 04:25 Public

当我们在某个目录下执行，看到有类似 drwxr-xr-x，这样的文件就是目录，目录在 Linux 是一个比较特殊的文件。注意它的第一个字符是 d。创建目录的命令可以用 mkdir 命令，或 cp 命令，cp 可以把一个目录复制为另一个目录。删除用 rm 或 rmdir 命令。

3 字符设备或块设备文件

如时您进入/dev 目录，列一下文件，会看到类似如下的：

1. [root@localhost ~]# ls -la /dev/tty
2. crw-rw-rw- 1 root tty 5, 0 04-19 08:29 /dev/tty
3. [root@localhost ~]# ls -la /dev/hda1
4. brw-r----- 1 root disk 3, 1 2006-04-19 /dev/hda1

我们看到/dev/tty 的属性是 crw-rw-rw-，注意前面第一个字符是 c，这表示字符设备文件。比如猫等串口设备

我们看到 /dev/hda1 的属性是 brw-r-----，注意前面的第一个字符是 b，这表示块设备，比如硬盘，光驱等设备；

这个种类的文件，是用 mknode 来创建，用 rm 来删除。目前在最新的 Linux 发行版本中，我们一般不用自己来创建设备文件。因为这些文件是和内核相关联的。

4 套接口文件

当我们启动 MySQL 服务器时，会产生一个 mysql.sock 的文件。

1. [root@localhost ~]# ls -lh /var/lib/mysql/mysql.sock
2. srwxrwxrwx 1 mysql mysql 0 04-19 11:12 /var/lib/mysql/mysql.sock

注意这个文件的属性的第一个字符是 s。我们了解一下就行了。

5 符号链接文件

1. [root@localhost ~]# ls -lh setup.log
2. lrwxrwxrwx 1 root root 11 04-19 11:18 setup.log -> install.log

当我们查看文件属性时，会看到有类似 lrwxrwxrwx,注意第一个字符是 l，这类文件是链接文件。是通过 ln -s 源文件名 新文件名。上面是一个例子，表示 setup.log 是 install.log 的软链接文件。怎么理解呢？这和 Windows 操作系统中的快捷方式有点相似。

符号链接文件的创建方法举例：

1. [root@localhost ~]# ls -lh kernel-6.15-1.2025_FC5.i686.rpm
2. -rw-r--r-- 1 root root 14M 03-16 07:53 kernel-6.15-1.2025_FC5.i686.rpm
3. [root@localhost ~]# ln -s kernel-6.15-1.2025_FC5.i686.rpm kernel.rpm
4. [root@localhost ~]# ls -lh kernel*
5. -rw-r--r-- 1 root root 14M 03-16 07:53 kernel-6.15-1.2025_FC5.i686.rpm
6. lrwxrwxrwx 1 root root 33 04-19 11:27 kernel.rpm -> kernel-6.15-1.2025_FC5

2、文件权限

Linux 系统是一个典型的多用户系统，不同的用户处于不同的地位。为了保护系统的安全性，Linux 系统对不同用户访问同一文件的权限做了不同的规定。

对于一个 Linux 系统中的文件来说，它的权限可以分为三种：读的权限、写的权限和执行的权限，分别用 r、w 和 x 表示。不同的用户具有不同的读、写和执行的权限。

对于一个文件来说，它都有一个特定的所有者，也就是对文件具有所有权的用户。同时，由于在 Linux 系统中，用户是按组分类的，一个用户属于一个或多个组。文件所有者以外的用户又可以分为文件所有者的同组用户和其它用户。因此，Linux 系统按文件所有者、文件所有者同组用户和其它用户三类规定不同的文件访问权限。

权限的概念

Linux 文件系统安全模型是通过给系统中的文件赋予两个属性来起作用的，这两个赋予每个文件的属性称为所有者(ownership)和访问权限(access rights)。Linux 下的每一个文件必须严格地属于一个用户和一个组。

下图是在我机器上的/root 目录下运行 ls -l 命令的情况。


```
[root@RedHat ~]# 11
总计 80
-rw----- 1 root root 1460 03-04 05:03 anaconda-ks.cfg
drwxr-xr-x 2 root root 4096 03-04 06:01 Desktop
-rw-r--r-- 1 root root 47018 03-04 05:03 install.log
-rw-r--r-- 1 root root 4186 03-04 05:01 install.log.syslog
drwxr-xr-x 7 root root 4096 01-23 09:25 vmware-tools-distrib
```

```
-rw-r--r--
```

这些符号用来描述文件的访问权限类别，也就是常说的文件权限。这些访问权限指导 Linux 根据文件的用户和组所有权来处理所有访问文件的用户请求。总共有 10 种权限属性，因此一个权限列表总是 10 个字符的长度。它的格式遵循下列规则：

◆ 第 1 个字符表示一种特殊的文件类型。其中字符可为 **d**(表示该文件是一个目录)、**b**(表示该文件是一个系统设备，使用块输入/输出与外界交互，通常为一个磁盘)、**c**(表示该文件是一个系统设备，使用连续的字符输入/输出与外界交互，如串口和声音设备)，**“.”** 表示该文件是一个普通文件，没有特殊属性。

◆ 2~4 个字符用来确定文件的用户(user)权限，5~7 个字符用来确定文件的组(group)权限，8~10 个字符用来确定文件的其它用户(other user，既不是文件所有者，也不是组成员的用户)的权限。其中，2、5、8 个字符是用来控制文件的读权限的，该位字符为 **r** 表示允许用户、组成员或其它人可从该文件中读取数据。短线“-”则表示不允许该成员读取数据。与此类似，3、6、9 位的字符控制文件的写权限，该位若为 **w** 表示允许写，若为“-”表示不允许写。4、7、10 位的字符用来控制文件的制造权限，该位若为 **x** 表示允许执行，若为“-”表示不允许执行。

任何列在 `/etc/passwd` 文件中的用户都可以是一个文件的所有者，也称为该文件的用户。同样任何列在 `/etc/group` 文件中的组都可以是文件组的所有者，也简称为文件的组。

```
drwxrwxr-- 2 root root 4096 2 月 11 10:36 guo
```

因为 `guo` 的第 1 个位置的字符是 **d**，所以由此知道 `guo` 是一个目录。第 2 至 4 位置上的属性是 **rw**，表示用户 `root` 拥有权限列表显示 `guo` 中所有的文件、创建新文件或者删除 `guo` 中现有的文件，或者将 `guo` 作为当前工作目录。第 5 至 7 个位置上的权限是 **rw**，表示 `root` 组的成员拥有和 `root` 一样的权限。第 8 至 10 位上的权限仅是 **r--**，表示不是 `root` 的用户及不属于 `root` 组的成员只有对 `guo` 目录列表的权限。这些用户不能创建或者删除 `guo` 中的文件、执行 `junk` 中的可执行文件，或者将 `junk` 作为他们的当前工作目录。

```
-rwxr-xr-- 1 user admin 20480 11 月 11 09:23 Readme.txt
```

在该项中，第 1 个位置是短线“-”，表示该文件是一个普通文件，没有特殊属性。该文件对任何人都可读，只对 `user` 可写，`user` 和 `admin` 的组成员可以执行该文件。

另外需要注意的是，当用户访问一个文件时，权限检查是从左到右的。假设上述的 `readme.txt` 文件具有以下权限：

```
-r--rw-r--
```

那么即使 `user` 是属于 `admin` 组的一个成员，也不能对该文件进行写操作，因为已经被左边的写权限设置拒绝了。

一般文件权限读 (R)，写(W)，执行 (X) 权限比较简单。一般材料上面都有介绍。这里介绍一下一些特殊的文件权限——SUID，SGID，Stick bit。

如果你检查一下 `/usr/bin/passwd` 和 `/tmp/` 的文件权限你就会发现和普

通的文件权限有少许不同，如下图所示：

```
simon@simon-laptop:~$ ls -l /usr/bin/passwd;ls -ld /tmp/
-rwsr-xr-x 1 root root 41292 2009-07-31 21:55 /usr/bin/passwd
drwxrwxrwt 7 root root 4096 2010-03-27 22:30 /tmp/
```

这里就涉及到 SUID 和 Stick bit。

SUID 和 SGID

我们首先来谈一下 passwd 程序特殊的地方。大家都知道，Linux 把用户的密码信息存放在 /etc/shadow 里面，该文件属性如下：

```
simon@simon-laptop:~$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1093 2010-01-02 21:15 /etc/shadow
```

可以看到 Shadow

的只有所有者可读写，所有者是 root，所以该文件对普通用户是不可读写的。但是普通用户调用 passwd 程序是可以修改自己的密码的，这又是为什么呢？难道普通用户可以读写 shadow 文件？难道 Linux 有漏洞？当然不是啦。password 可以修改 shadow 文件的原因是他设置了 SUID 文件权限。

SUID 文件权限作用于可执行文件。一般的可执行文件在执行期的所有者是当前用户，比如当前系统用户是 simon，simon 运行程序 a.out，a.out 执行期的所有者应该是 simon。但是如果给可执行文件设置了 SUID 权限，则该程序的执行期所有者，就是该文件所有者。还以 前面的 a.out 为例，假如 a.out 设置了 SUID，并且其所有者是 root，系统当前用户是 simon，当 simon 运行 a.out 的时 候，a.out 在运行期的所有者就是 root，这时 a.out 可以存取只有 root 权限才能存取的资源，比如读写 shadow 文件。当 a.out 执行结束 的时候当前用户的权限又回到了 simon 的权限了。

passwd 就是设置了 SUID 权限，并且 passwd 的所有者是 root，所以所有的用户都可以执行他，在 passwd 运行期，程序获得临时的 root 权限，这时其可以存取 shadow 文件。当 passwd 运行完成，当前用户又回到普通权限。

同理，设置程序的 SGID，可以使程序运行期可以临时获得所有者组的权限。在团队开发的时候，这个文件权限比较有用，一般系统用 SUID 比较多。

SGID 可以用于目录，当目录设置了 SGID 之后，在该目录下面建立的所有文件和目录都具有和该目录相同的用户组。

Stick bit(粘贴位)

对程序，该权限告诉系统在程序完成后在内存中保存一份运行程序的备份，如该程序常用，可为系统节省点时间，不用每次从磁盘加载到内存。Linux 当前对文件没有实现这个功能，一些其他的 UNIX 系统实现了这个功能。

Stick bit 可以作用于目录，在设置了粘贴位的目录下面的文件和目录，只有所有者和 root 可以删除他。现在我们可以回头去看看 /tmp/ 目录的情况，这个目录 设置了粘贴位。所以说，并且所有人都可以对该目录读写执行（777），这意味着所有人都可以在 /tmp/ 下面创建临时

目录。因为设置 Stick bit 只有所有者和 root 才能删除目录。这样普通用户只能删除属于自己的文件，而不能删除其他人的文件。如下图所示：

```
simon@simon-laptop:/tmp$ sudo touch root_file;sudo chmod 777 root_file;ls -l root_file
-rwxrwxrwx 1 root root 0 2010-03-27 23:29 root_file
simon@simon-laptop:/tmp$ rm root file
rm: 无法删除 "root_file": 操作不允许
```

设置 SUID, SGID, Sticky bit

前面介绍过 SUID 与 SGID 的功能，那么，如何打开文件使其成为具有 SUID 与 SGID 的权限呢？这就需要使用数字更改权限了。现在应该知道，使用数字 更改权限的方式为“3 个数字”的组合，那么，如果在这 3 个数字之前再加上一个数字，最前面的数字就表示这几个属性了（注：通常我们使用 `chmod xyz filename` 的方式来设置 `filename` 的属性时，则是假设没有 SUID、SGID 及 Sticky bit）。

4 为 SUID

2 为 SGID

1 为 Sticky bit

假设要将一个文件属性改为“-rwsr-xr-x”，由于 s 在用户权限中，所以是 SUID，因此，在原先的 755 之前还要加上 4，也就是使用“`chmod 4755 filename`”来设置。

SUID 也可以用“`chmod u+s filename`”来设置，“`chmod u-s filename`”来取消 SUID 设置；同样，SGID 可以用“`chmod g+s filename`”，“`chmod g-s filename`”来取消 SGID 设置。

一般来说，使用过 Linux 的同学都知道，Linux 文件的权限有 rwx，所有者、所有组、其它用户的 rwx 权限是彼此独立的。为此，经常会听到如果某个 web 文件需要被修改的话，需要加上 777 的权限，这就是让所有用户可写。

但仔细一想，这样的权限未免有些想得比较天真，没有考虑特殊情况。例如/tmp 目录默认权限是 777，而且有些文件也是允许所有用户访问修改的，那么是不是任何一个用户都可以将这些删除呢？再如/etc/shadow 保存的是用户密码文件，默认情况下它的权限是 640，那么只有 shadow 的 owner(root)才能修改它，按照常规理解，这是不可理解的，因为每个用户都可能修改密码，也就是会修改这个文件。

为了把这些情况解释清楚，需要引入 Linux 特殊文件权限的概念。Linux 特殊文件权限有三个玩意：sticky bit、SGID、SUID，以下一一道来。

sticky bit

sticky bit 只对目录有效，使目录下的文件，只有文件拥有者才能删除（如果他不属于 owner，仅属于 group 或者 other，就算他有 w 权限，也不能删除文件）。

加 sticky bit 的方法：

`chmod o+t /tmp` 或者

```
chmod 1777 /tmp
```

查看是否加了 sticky bit，用 `ls -l`，可以看到有类似这样的权限：“-rwxrwxrwt”，t 就代表已经加上了 sticky bit，而且生效了，如果显示的是“-rwxrwxrwt”，说明也已经加上了 sticky bit，但没有生效（因为本来 other 就没有写的权限）。

看看/tmp 目录的权限，就是 drwxrwxrwt 吧

SGID(The Set GroupID)

加上 SGID 的文件，表示运行这个程序时，是临时以这个文件的拥有组的身份运行的；加上 SGID 的文件夹，表示在这个目录下创建的文件属于目录所有的组，而不是创建人所在的组，在这个目录下创建的目录继承本目录的 SGID。

加 SGID 的方法：

```
chmod g+s /tmp 或
```

```
chmod 2777 /tmp
```

查看是否加了 SGID，用 `ls -l`，可以看到类似这样的权限“drwxrwsrwx”，s 就代表已经加上了 SGID，而且生效，如果显示“drwxrwSrwx”，说明已经加上了 SGID，但没有生效（因为本来 group 就没有执行的权限）。

SUID(The Set UserID)

SUID 与 SGID 是一样的，惟一不同的是，运行时是以这个文件的拥有者身份来运行。

加 SUID 的方法：

```
chmod o+s /tmp 或
```

```
chmod 4777 /tmp
```

同样的，加了 SUID 的文件权限有这类似这两种：“drwsrwxrwx”、“drwSrwxrwx”。

看看 `passwd` 命令的权限：`ll /usr/bin/passwd`，是“-rwsr-x-rx”，终于知道为什么执行 `passwd` 时，可以修改/etc/shadow 文件了吧

SUID：置于 u 的 x 位，原位置有执行权限，就置为 s，没有了为 S。

SGID：置于 g 的 x 位，原位置有执行权限，就置为 s，没有了为 S。

STICKY：粘滞位，置于 o 的 x 位，原位置有执行权限，就置为 t，否则为 T。

3、超级权限控制

在 Linux 操作系统中，root 的权限是最高的，也被称为超级权限的拥有者。普通用户无法执行的操作，root 用户都能完成，所以也被称之为超级管理用户。

在系统中，每个文件、目录和进程，都归属于某一个用户，没有用户许可其它普通用户是无法操作的，但对 root 除外。root 用户的特权性还表现在 root 可以超越任何用户和用户组来对文件或目录进行读取、修改或删除（在系统正常的许可范围内）；对可执行程序的执行、终止；对硬件设备的添加、创建和移除等；也可以对文件和目录进行属主和权限进行修改，以适合系统管理的需要（因为 root 是系统中权限最高的特权用户）；

3.1、对超级用户和普通用户的理解

3.1.1、什么是超级用户；

在所有 Linux 系统中，系统都是通过 UID 来区分用户权限级别的，而 UID 为 0 的用户被系统约定为是具有超级权限。超级用户具有在系统约定的最高权限满园内操作，所以说超级用户可以完成系统管理的所有工具；我们可以通过/etc/passwd 来查得 UID 为 0 的用户是 root，而且只有 root 对应的 UID 为 0，从这一点来看，root 用户在系统中是无可替代的至高地位和无限制权限。root 用户在系统中就是超级用户；

3.1.2、理解 UID 和用户的对应关系

当系统默认安装时，系统用户和 UID 是一一对应的关系，也就是说一个 UID 对应一个用户。我们知道用户身份是通过 UID 来确认的，我们在《用户（user）和用户组（group）配置文件详解》中的 UID 的解说中有谈到“UID 是确认用户权限的标识，用户登录系统所处的角色是通过 UID 来实现的，而非用户名；把几个用户共用一个 UID 是危险的，比如我们把普通用户的 UID 改为 0，和 root 共用一个 UID，这事实上就造成了系统管理权限的混乱。如果我们想用 root 权限，可以通过 su 或 sudo 来实现；切不可随意让一个用户和 root 分享同一个 UID；”

在系统中，能不能让 UID 和用户是一对多的关系？是可以的，比如我们可以把一个 UID 为 0 这个值分配给几个用户共同使用，这就是 UID 和用户的一对多的关系。但这样做的确有点危险；相同 UID 的用户具有相同的身份和权限。比如我们在系统中把 beinan 这个普通用户的 UID 改为 0 后，事实上这个普通用户就具有了超级权限，他的能力和权限和 root 用户一样；用户 beinan 所有的操作都将被标识为 root 的操作，因为 beinan 的 UID 为 0，而 UID 为 0 的用户是 root，是不是有点扰口？也可以理解为 UID 为 0 的用户就是 root，root 用户的 UID 就是 0；

UID 和用户的一一对应的关系，只是要求管理员进行系统管理时，所要坚守的准则，因为系统安全还是第一位的。所以我们还是把超级权限保留给 root 这唯一的用户是最好的选择；

如果我们不把 UID 的 0 值的分享给其它用户使用，只有 root 用户是唯一拥有 UID=0 的话，root 用户就是唯一的超级权限用户；

3.1.3、普通用户和伪装用户

与超级用户相对的就是普通用户和虚拟（也被称为伪装用户），普通和伪装用户都是受限用户；但为了完成特定的任务，普通用户和伪装用户也是必须的；Linux 是一个多用户、多任务的操作系统，多用户主要体现在用户的角色的多样性，不同的用户所分配的权限也不同；这也是 Linux 系统比 Windows 系统更为安全的本质所在，即使是现在最新版本的 Windows 2003，也无法抹去其单用户系统的烙印；

3.2. 超级用户（权限）在系统管理中的作用

超级权限用户（UID 为 0 的用户）到底在系统管理中起什么作用呢？主要表现在以下两点；

3.2.1、对任何文件、目录或进程进行操作；

但值得注意的是这种操作是在系统最高许可范围内的操作；有些操作就是具有超级权限的 root 也无法完成；

比如 /proc 目录，/proc 是用来反应系统运行的实时状态信息的，因此即便是 root 也无能为力；它的权限如下

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# cd /
[root@localhost /]# ls -ld /proc/
dr-xr-xr-x 134 root root 0 2005-10-27 /proc/
```

就是这个目录，只能是读和执行权限，但绝对没有写权限的；就是我们把/proc 目录的写权限打开给 root，root 用户也是不能进行写操作；

```
[root@localhost ~]# chmod 755 /proc
[root@localhost /]# ls -ld /proc/
drwxr-xr-x 134 root root 0 2005-10-27 /proc/
[root@localhost /]# cd /proc/
[root@localhost proc]# mkdir testdir
mkdir: 无法创建目录 'testdir'：没有那个文件或目录
```

3.2.2、对于涉及系统全局的系统管理；

硬件管理、文件系统理解、用户管理以及涉及到的系统全局配置等等.....如果您执行某个命令或工具时，提示您无权限，大多是需要超级权限来完成；

比如用 **adduser** 来添加用户，这个只能用通过超级权限的用户来完成；

3.2.3、超级权限的不可替代性；

由于超级权限在系统管理中的不可缺少的重要作用，为了完成系统管理任务，我们必须用到超级权限；在一般情况下，为了系统安全，对于一般常规级别的应用，不需要 root 用户来操作完成，root 用户只是被用来管理和维护系统之用；比如系统日志的查看、清理，用户的添加和删除.....

在不涉及系统管理的工作的环境下，普通用户是可以完成，比如编写一个文件，听听音乐；用 **gimp** 处理一个图片等..... 基于普通应用程序的调用，大多普通用户就可以完成；

当我们以普通权限的用户登录系统时，有些系统配置及系统管理必须通过超级权限用户完成，比如对系统日志的管理，添加和删除用户。而如何才能不直接以 root 登录，却能从普通用户切换到 root 用户下才能进行操作系统管理需要的工作，这就涉及到超级权限管理的问题；

获取超级权限的过程，就是切换普通用户身份到超级用户身份的过程；这个过程主要是通过 **su** 和 **sudo** 来解决；

3.3、使用 su 命令临时切换用户身份；

3.3.1、su 的适用条件和威力

su 命令就是切换用户的工具，怎么理解呢？比如我们以普通用户 **beinan** 登录的，但要添加用户任务，执行 **useradd**，**beinan** 用户没有这个权限，而这个权限恰恰由 root 所拥有。解决办法无法有两个，一是退出 **beinan** 用户，重新以 root 用户登录，但这种办法并不是最好的；二是我们没有必要退出 **beinan** 用户，可以用 **su** 来切换到 root 下进行添加用户的工作，等任务完成后再退出 root。我们可以看到当然通过 **su** 切换是一种比较好的办法；

通过 **su** 可以在用户之间切换，如果超级权限用户 root 向普通或虚拟用户切换不需要密码，什么是权力？这就是！而普通用户切换到其它任何用户都需要密码验证；

3.3.2、su 的用法：

```
su [OPTION 选项参数] [用户]
-, -l, --login          登录并改变到所切换的用户环境；
```


`-c, --command=COMMAND` 执行一个命令，然后退出所切换到的用户环境；

至于更详细的，请参看 `man su` ；

3.3.3、su 的范例：

`su` 在不加任何参数，默认为切换到 `root` 用户，但没有转到 `root` 用户家目录下，也就是说这时虽然是切换为 `root` 用户了，但并没有改变 `root` 登录环境；用户默认的登录环境，可以在 `/etc/passwd` 中查得到，包括家目录，`SHELL` 定义等；

```
[beinan@localhost ~]$ su
Password:
[root@localhost beinan]# pwd
/home/beinan
```

`su` 加参数 `-`，表示默认切换到 `root` 用户，并且改变到 `root` 用户的环境；

```
[beinan@localhost ~]$ pwd
/home/beinan
[beinan@localhost ~]$ su -
Password:
[root@localhost ~]# pwd
/root
```

`su` 参数 `-` 用户名

```
[beinan@localhost ~]$ su - root 注：这个和 su - 是一样的功能；
Password:
[root@localhost ~]# pwd
/root
[beinan@localhost ~]$ su - linuxsir 注：这是切换到 linuxsir 用户
Password: 注：在这里输入密码；
[linuxsir@localhost ~]$ pwd 注：查看用户当前所处的位置；
/home/linuxsir
[linuxsir@localhost ~]$ id 注：查看用户的 UID 和 GID 信息，主要是看是否切换过来了；
uid=505(linuxsir) gid=502(linuxsir) groups=0(root), 500(beinan), 502(linuxsir)
[linuxsir@localhost ~]$
[beinan@localhost ~]$ su - -c ls 注：这是 su 的参数组合，表示切换到 root 用户，并且改变到 root 环境，然后列出 root 家目录的文件，然后退出 root 用户；
Password: 注：在这里输入 root 的密码；
anaconda-ks.cfg Desktop install.log install.log.syslog testgroup testgroupbeinan testgrouproot
[beinan@localhost ~]$ pwd 注：查看当前用户所处的位置；
/home/beinan
[beinan@localhost ~]$ id 注：查看当前用户信息；
uid=500(beinan) gid=500(beinan) groups=500(beinan)
```

3.3.4、su 的优缺点；

`su` 的确为管理带来方便，通过切换到 `root` 下，能完成所有系统管理工具，只要把 `root` 的密码交给任何一个普通用户，他都能切换到 `root` 来完成所有的系统管理工作；

但通过 `su` 切换到 `root` 后，也有不安全因素；比如系统有 10 个用户，而且都参与管理。如果这 10 个用户都涉及到超级权限的运用，做为管理员如果想让其它用户通过 `su` 来切换到超级权限的 `root`，必须把 `root` 权限密码都告诉这 10 个用户；如果这

10 个用户都有 root 权限，通过 root 权限可以做任何事，这在一定程度上就对系统的安全造成了威胁；想想 Windows 吧，简直就是恶梦；

“没有不安全的系统，只有不安全的人”，我们绝对不能保证这 10 个用户都能按正常操作流程来管理系统，其中任何一人对系统操作的重大失误，都可能导致系统崩溃或数据损失；

所以 su 工具在多人参与的系统管理中，并不是最好的选择，su 只适用于一两个人参与管理的系统，毕竟 su 并不能让普通用户受限的使用；

超级用户 root 密码应该掌握在少数用户手中，这绝对是真理！所以集权而治的存在还是有一定道理的；

3.4、sudo 授权许可使用的su，也是受限制的su

3.4.1. sudo 的适用条件；

由于 su 对切换到超级权限用户 root 后，权限的无限制性，所以 su 并不能担任多个管理员所管理的系统。如果用 su 来切换到超级用户来管理系统，也不能明确哪些工作是由哪个管理员进行的操作。特别是对于服务器的管理有多人参与管理时，最好是针对每个管理员的技术特长和管理范围，并且有针对性的下放给权限，并且约定其使用哪些工具来完成与其相关的工作，这时我们就有必要用到 sudo。

通过 sudo，我们能将某些超级权限有针对性的下放，并且不需要普通用户知道 root 密码，所以 sudo 相对于权限无限制性的 su 来说，还是比较安全的，所以 sudo 也能被称为受限制的 su；另外 sudo 是需要授权许可的，所以也被称为授权许可的 su；

sudo 执行命令的流程是当前用户切换到 root（或其它指定切换到的用户），然后以 root（或其它指定的切换到的用户）身份执行命令，执行完成后，直接退回到当前用户；而这些的前提是要通过 sudo 的配置文件/etc/sudoers 来进行授权；

3.4.2、从编写 sudo 配置文件/etc/sudoers 开始；

sudo 的配置文件是/etc/sudoers，我们可以用他的专用编辑工具 visodu，此工具的好处是在添加规则不太准确时，保存退出时会提示给我们错误信息；配置好后，可以用切换到您授权的用户下，通过 sudo -l 来查看哪些命令是可以执行或禁止的；

/etc/sudoers 文件中每行算一个规则，前面带有#号可以当作是说明的内容，并不执行；如果规则很长，一行列不下时，可以用\号来续行，这样看来一个规则也可以拥有多个行；

/etc/sudoers 的规则可分为两类；一类是别名定义，另一类是授权规则；别名定义并不是必须的，但授权规则是必须的；

3.4.3、/etc/sudoers 配置文件中别名规则

别名规则定义格式如下：

```
Alias_Type NAME = item1, item2, ...
```

或

```
Alias_Type NAME = item1, item2, item3 : NAME = item4, item5
```

别名类型（Alias_Type）：别名类型包括如下四种

Host_Alias 定义主机别名；

User_Alias 用户别名，别名成员可以是用户，用户组（前面要加%号）

Runas_Alias 用来定义 runas 别名，这个别名指定的是“目的用户”，即 sudo 允许切换至的用户；

Cmnd_Alias 定义命令别名；

NAME 就是别名了，NAME 的命名是包含大写字母、下划线以及数字，但必须以一个大写字母开头，比如 SYNADM、SYN_ADM 或 SYNAD0 是合法的，sYNAMDA 或 1SYNAD 是不合法的；

item 按中文翻译是项目，在这里我们可以译成成员，如果一个别名下有多个成员，成员与成员之间，通过半角,号分隔；成员在必须是有效并事实存在的。什么是有效的呢？比如主机名，可以通过 **w** 查看用户的主机名（或 **ip** 地址），如果您只是本地机操作，只通过 **hostname** 命令就能查看；用户名当然是在系统中存在的，在 **/etc/passwd** 中必须存在；对于定义命令别名，成员也必须在系统中事实存在的文件名（需要绝对路径）；

item 成员受别名类型 **Host_Alias**、**User_Alias**、**Runas_Alias**、**Cmnd_Alias** 制约，定义什么类型的别名，就要有什么类型的成员相配。我们用 **Host_Alias** 定义主机别名时，成员必须是与主机相关相关联，比如是主机名（包括远程登录的主机名）、**ip** 地址（单个或整段）、掩码等；当用户登录时，可以通过 **w** 命令来查看登录用户主机信息；用 **User_Alias** 和 **Runas_Alias** 定义时，必须要用系统用户做为成员；用 **Cmnd_Alias** 定义执行命令的别名时，必须是系统存在的文件，文件名可以用通配符表示，配置 **Cmnd_Alias** 时命令需要绝对路径；

其中 **Runas_Alias** 和 **User_Alias** 有点相似，但与 **User_Alias** 绝对不是同一个概念，**Runas_Alias** 定义的是某个系统用户可以 **sudo** 切换身份到 **Runas_Alias** 下的成员；我们在授权规则中以实例进行解说；

别名规则是每行算一个规则，如果一个别名规则一行容不下时，可以通过\来续行；同一类型别名的定义，一次也可以定义几个别名，他们中间用:号分隔，

```
Host_Alias    HT01=localhost, st05, st04, 10, 0, 0, 4, 255. 255. 255. 0, 192. 168. 1. 0/24    注：定义主机别名 HT01，通过=号列出成员
Host_Alias    HT02=st09, st10    注：主机别名 HT02，有两个成员；
Host_Alias    HT01=localhost, st05, st04, 10, 0, 0, 4, 255. 255. 255. 0, 192. 168. 1. 0/24:HT02=st09, st10    注：上面的两条对主机的定义，可以通过一条来实现，别名之间用:号分割；
```

注：我们通过 **Host_Alias** 定义主机别名时，项目可以是主机名、可以是单个 **ip**（整段 **ip** 地址也可以），也可以是网络掩码；如果是主机名，必须是多台机器的网络中，而且这些机器得能通过主机名相互通信访问才有效。那什么才算是通过主机名相互通信或访问呢？比如 **ping** 主机名，或通过远程访问主机名来访问。在我们局域网中，如果让计算机通过主机名访问通信，必须设置 **/etc/hosts**，**/etc/resolv.conf**，还要有 **DNS** 做解析，否则相互之间无法通过主机名访问；在设置主机别名时，如果项目是中某个项目是主机名的话，可以通过 **hostname** 命令来查看本地主机的主机名，通过 **w** 命令查来看登录主机是来源，通过来源来确认其它客户机的主机名或 **ip** 地址；对于主机别名的定义，看上去有点复杂，其实是很简单。

如果您不明白 **Host_Alias** 是怎么回事，也可以不用设置主机别名，在定义授权规则时通过 **ALL** 来匹配所有可能出现的主机情况。如果您把主机方面的知识弄的更明白，的确需要多多学习。

```
User_Alias    SYSAD=beinan, linuxsir, bnnnb, lanhaitun    注：定义用户别名，下有四个成员；要在系统中确实存在的；
User_Alias    NETAD=beinan, bnnb    注：定义用户别名 NETAD，我想让这个别名下的用户来管理网络，所以取了 NETAD 的别名；
User_Alias    WEBMASTER=linuxsir    注：定义用户别名 WEBMASTER，我想用这个别名下的用户来管理网站；
User_Alias    SYSAD=beinan, linuxsir, bnnnb, lanhaitun:NETAD=beinan, bnnb:WEBMASTER=linuxsir    注：上面三行的别名定义，可以通过这一行来实现，请看前面的说明，是不是符合？
Cmnd_Alias    USERMAG=/usr/sbin/adduser, /usr/sbin/userdel, /usr/bin/passwd [A-Za-z]*, /bin/chown, /bin/chmod
注意：命令别名下的成员必须是文件或目录的绝对路径；
Cmnd_Alias    DISKMAG=/sbin/fdisk, /sbin/parted
Cmnd_Alias    NETMAG=/sbin/ifconfig, /etc/init.d/network
Cmnd_Alias    KILL = /usr/bin/kill
Cmnd_Alias    PWMAG = /usr/sbin/reboot, /usr/sbin/halt
Cmnd_Alias    SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                    /usr/local/bin/tcsh, /usr/bin/rsh, \
                    /usr/local/bin/zsh
```

注：这行定义命令别名有点长，可以通过 \ 号断行；

```
Cmnd_Alias    SU = /usr/bin/su, /bin, /sbin, /usr/sbin, /usr/bin
```

在上面的例子中，有 **KILL** 和 **PWMAG** 的命令别名定义，我们可以合并为一行来写，也就是等价行：

```
Cmnd_Alias    KILL = /usr/bin/kill:PWMAG = /usr/sbin/reboot, /usr/sbin/halt  注：这一行就代表了 KILL 和 PWMAG
命令别名，把 KILL 和 PWMAG 的别名定义合并在一行写也是可以的；
```

```
Runas_Alias    OP = root, operator
```

```
Runas_Alias    DBADM=mysql:OP = root, operator  注：这行是上面两行的等价行；至于怎么理解 Runas_Alias，我们必须
得通过授权规则的实例来理解；
```

3.4.4、/etc/sudoers 中的授权规则：

授权规则是分配权限的执行规则，我们前面所讲到的定义别名主要是为了更方便的授权引用别名；如果系统中只有几个用户，其实下放权限比较有限的话，可以不用定义别名，而是针对系统用户直接直接授权，所以在授权规则中别名并不是必须的；

授权规则并不是无章可寻，我们只说基础一点的，比较简单的写法，如果您想详细了解授权规则写法的，请参看 **man sudoers**

授权用户 主机=命令动作

这三个要素缺一不可，但在动作之前也可以指定切换到特定用户下，在这里指定切换的用户要用()号括起来，如果不需要密码直接运行命令的，应该加 **NOPASSWD:**参数，但这些可以省略；举例说明：

实例一：

```
beinan ALL=/bin/chown, /bin/chmod
```

如果我们在 **/etc/sudoers** 中添加这一行，表示 **beinan** 可以在任何可能出现的主机名的系统中，可以切换到 **root** 用户下执行 **/bin/chown** 和 **/bin/chmod** 命令，通过 **sudo -l** 来查看 **beinan** 在这台主机上允许和禁止运行的命令；

值得注意的是，在这里省略了指定切换到哪个用户下执行 **/bin/chown** 和 **/bin/chmod** 命令；在省略的情况下默认是切换到 **root** 用户下执行；同时也省略了是不是需要 **beinan** 用户输入验证密码，如果省略了，默认是需要验证密码。

为了更详细的说明这些，我们可以构造一个更复杂一点的公式：

授权用户 主机=[(切换到哪些用户或用户组)] [是否需要密码验证] 命令 1,[(切换到哪些用户或用户组)] [是否需要密码验证]
[命令 2],[切换到哪些用户或用户组)] [是否需要密码验证] [命令 3].....

注解：

凡是[]中的内容，是可以省略；命令与命令之间用,号分隔；通过本文的例子，可以对照着看哪些是省略了，哪些地方需要有空格；

在[(切换到哪些用户或用户组)]，如果省略，则默认为 **root** 用户；如果是 **ALL**，则代表能切换到所有用户；注意要切换到的目的用户必须用()号括起来，比如(**ALL**)、(**beinan**)

实例二：

```
beinan ALL=(root) /bin/chown, /bin/chmod
```

如果我们把第一个实例中的那行去掉，换成这行；表示的是 **beinan** 可以在任何可能出现的主机名的主机中，可以切换到 **root** 下执行 **/bin/chown**，可以切换到任何用户招执行 **/bin/chmod** 命令，通过 **sudo -l** 来查看 **beinan** 在这台主机上允许和禁止运行的命令；

实例三：

```
beinan ALL=(root) NOPASSWD: /bin/chown, /bin/chmod
```

如果换成这个例子呢？表示的是 **beinan** 可以在任何可能出现的主机名的主机中，可以切换到 **root** 下执行 **/bin/chown**，不需要输入 **beinan** 用户的密码；并且可以切换到任何用户下执行 **/bin/chmod** 命令，但执行 **chmod** 时需要 **beinan** 输入自己的密码；通过 **sudo -l** 来查看 **beinan** 在这台主机上允许和禁止运行的命令；

关于一个命令动作是不是需要密码，我们可以发现在系统在默认的情况下是需要用户密码的，除非特加指出不需要用户需要输入自己密码，所以要在执行动作之前加入 **NOPASSWD:** 参数；

有可能有的弟兄对系统管理的命令不太懂，不知道其用法，这样就影响了他对 **sudoers** 定义的理解，下面我们再举一个最简单，最有说服务力的例子；

实例四：

比如我们想用 **beinan** 普通用户通过 **more /etc/shadow** 文件的内容时，可能会出现下面的情况；

```
[beinan@localhost ~]$ more /etc/shadow
/etc/shadow: 权限不够
```

这时我们可以用 **sudo more /etc/shadow** 来读取文件的内容；就就需要在 **/etc/soduers** 中给 **beinan** 授权；

于是我们就可以先 **su** 到 **root** 用户下通过 **visudo** 来改 **/etc/sudoers**；（比如我们是以 **beinan** 用户登录系统的）

```
[beinan@localhost ~]$ su
Password: 注：在这里输入 root 密码
下面运行 visodu;
[root@localhost beinan]# visudo 注：运行 visudo 来改 /etc/sudoers
```

加入如下一行，退出保存；退出保存，在这里要会用 **vi**，**visudo** 也是用的 **vi** 编辑器；至于 **vi** 的用法不多说了；

beinan ALL=/bin/more 表示 **beinan** 可以切换到 **root** 下执行 **more** 来查看文件；

退回到 **beinan** 用户下，用 **exit** 命令；

```
[root@localhost beinan]# exit
exit
[beinan@localhost ~]$
```

查看 **beinan** 的通过 **sudo** 能执行哪些命令？

```
[beinan@localhost ~]$ sudo -l
Password: 注：在这里输入 beinan 用户的密码
User beinan may run the following commands on this host: 注：在这里清晰的说明在本台主机上，beinan 用户可以以
root 权限运行 more；在 root 权限下的 more，可以查看任何文本文件的内容的；
(root) /bin/more
```

最后，我们看看是不是 **beinan** 用户有能力看到 **/etc/shadow** 文件的内容；

```
[beinan@localhost ~]$ sudo more /etc/shadow
```

beinan 不但能看到 **/etc/shadow** 文件的内容，还能看到只有 **root** 权限下才能看到的其它文件的内容，比如；

```
[beinan@localhost ~]$ sudo more /etc/gshadow
```

对于 **beinan** 用户查看和读取所有系统文件中，我只想吧 **/etc/shadow** 的内容可以让他查看；可以加入下面的一行；

```
beinan ALL=/bin/more /etc/shadow
```

题外话：有的弟兄会说，我通过 **su** 切换到 **root** 用户就能看到所有想看的内容了，哈哈，对啊。但咱们现在不是在讲述 **sudo** 的用法吗？如果主机上有多个用户并且不知道 **root** 用户的密码，但又想查看某些他们看不到的文件，这时就需要管理员授权了；这就是 **sudo** 的好处；

实例五：练习用户组在 **/etc/sudoers 中写法；**

如果用户组出现在`/etc/sudoers` 中，前面要加`%`号，比如`%beinan`，中间不能有空格；

```
%beinan  ALL=/usr/sbin/*,/sbin/*
```

如果我们在 `/etc/sudoers` 中加上如上一行，表示 `beinan` 用户组下的所有成员，在所有可能的出现的主机名下，都能切换到 `root` 用户下运行 `/usr/sbin` 和 `/sbin` 目录下的所有命令；

实例六：练习取消某类程序的执行；

取消程序某类程序的执行，要在命令动作前面加上`!`号； 在本例中也出现了通配符的`*`的用法；

```
beinan  ALL=/usr/sbin/*,/sbin/*,!/usr/sbin/fdisk    注：把这行规则加入到/etc/sudoers 中；但您得有 beinan 这个用户组，并且 beinan 也是这个组中的才行；
```

本规则表示 `beinan` 用户在所有可能存在的主机名的主机上运行`/usr/sbin` 和 `/sbin` 下所有的程序，但 `fdisk` 程序除外；

```
[beinan@localhost ~]$ sudo -l
```

Password: 注：在这里输入 `beinan` 用户的密码；

User beinan may run the following commands on this host:

```
(root) /usr/sbin/*
```

```
(root) /sbin/*
```

```
(root) !/sbin/fdisk
```

```
[beinan@localhost ~]$ sudo /sbin/fdisk -l
```

Sorry, user beinan is not allowed to execute '/sbin/fdisk -l' as root on localhost.

注：不能切换到 `root` 用户下运行 `fdisk` 程序；

实例七：别名的运用的实践；

假如我们就一台主机 `localhost`，能通过 `hostname` 来查看，我们在这里就不定义主机别名了，用 `ALL` 来匹配所有可能出现的主机名；并且有 `beinan`、`linuxsir`、`lanhaitun` 用户；主要是通过小例子能更好理解；`sudo` 虽然简单好用，但能把说的明白的确是件难事；最好的办法是多看例子和 `man sudoers` ；

```
User_Alias  SYSADER=beinan,linuxsir,%beinan
User_Alias  DISKADER=lanhaitun
Runas_Alias OP=root
Cmdnd_Alias SYDCMD=/bin/chown,/bin/chmod,/usr/sbin/adduser,/usr/bin/passwd [A-Za-z]*,!/usr/bin/passwd root
Cmdnd_Alias DSKCMD=/sbin/parted,/sbin/fdisk  注：定义命令别名 DSKCMD，下有成员 parted 和 fdisk ；
SYSADER     ALL= SYDCMD, DSKCMD
DISKADER     ALL=(OP)  DSKCMD
```

注解：

第一行：定义用户别名 `SYSADER` 下有成员 `beinan`、`linuxsir` 和 `beinan` 用户组下的成员，用户组前面必须加`%`号；

第二行：定义用户别名 `DISKADER`，成员有 `lanhaitun`

第三行：定义 `Runas` 用户，也就是目标用户的别名为 `OP`，下有成员 `root`

第四行：定义 `SYSCMD` 命令别名，成员之间用`,`号分隔，最后的`!/usr/bin/passwd root` 表示不能通过 `passwd` 来更改 `root` 密码；

第五行：定义命令别名 DSKCMD，下有成员 parted 和 fdisk ；

第六行：表示授权 SYSADER 下的所有成员，在所有可能存在的主机名的主机下运行或禁止 SYDCMD 和 DSKCMD 下定义的命令。更为明确地说，beinan、linuxsir 和 beinan 用户组下的成员能以 root 身份运行 chown 、chmod 、adduser、passwd，但不能更改 root 的密码；也可以以 root 身份运行 parted 和 fdisk ，本条规则的等价规则是：

```
beinan,linuxsir,%beinan  ALL=/bin/chown,/bin/chmod,/usr/sbin/adduser,/usr/bin/passwd
[A-Za-z]*,!/usr/bin/passwd root,/sbin/parted,/sbin/fdisk
```

第七行：表示授权 DISKADER 下的所有成员，能以 OP 的身份，来运行 DSKCMD ，不需要密码；更为明确的说 lanhaitun 能以 root 身份运行 parted 和 fdisk 命令；其等价规则是：

```
lanhaitun  ALL=(root) /sbin/parted,/sbin/fdisk
```

可能有的弟兄会说我想不输入用户的密码就能切换到 root 并运行 SYDCMD 和 DSKCMD 下的命令，那应该把把 NOPASSWD: 加在哪里为好？理解下面的例子吧，能明白的：

```
SYSADER  ALL= NOPASSWD: SYDCMD, NOPASSWD: DSKCMD
```

3.4.5、/etc/sudoers 中其它的未尽事项；

在授权规则中，还有 NOEXEC:和 EXEC 的用法，自己查 man sudoers 了解；还有关于在规则中通配符的用法，也是需要了解的。这些内容不多说了，毕竟只是一个入门性的文档。sudoers 配置文件要多简单就有多简单，要多难就有多难，就看自己的应用了。

3.4.6、sudo 的用法；

我们在前面讲的/etc/sudoers 的规则写法，最终的目的是让用户通过 sudo 读取配置文件中的规则来实现匹配和授权，以便替换身份来进行命令操作，进而完成在其权限下不可完成的任务；

我们只说最简单的用法；更为详细的请参考 man sudo

sudo [参数选项] 命令

-l 列出用户在主机上可用的和被禁止的命令；一般配置好/etc/sudoers 后，要用这个命令来查看和测试是不是配置正确的；

-v 验证用户的时间戳；如果用户运行 sudo 后，输入用户的密码后，在短时间内可以不用输入口令来直接进行 sudo 操作；用-v 可以跟踪最新的时间戳；

-u 指定以以某个用户执行特定操作；

-k 删除时间戳，下一个 sudo 命令要求用户提供密码；

举例：

首先我们通过 visudo 来改/etc/sudoers 文件，加入下面一行；

```
beinan,linuxsir,%beinan  ALL=/bin/chown,/bin/chmod,/usr/sbin/adduser,/usr/bin/passwd
[A-Za-z]*,!/usr/bin/passwd root,/sbin/parted,/sbin/fdisk
```

然后列出 beinan 用户在主机上通过 sudo 可以切换用户所能用的命令或被禁止用的命令；

```
[beinan@localhost ~]$ sudo -l 注：列出用户在主机上能通过切换用户的可用的或被禁止的命令；
```

Password: 注：在这里输入您的用户密码；

User beinan may run the following commands on this host:

(root) /bin/chown 注：可以切换到 root 下用 chown 命令；

(root) /bin/chmod 注：可以切换到 root 下用 chmod 命令；

(root) /usr/sbin/adduser 注：可以切换到 root 下用 adduser 命令；

(root) /usr/bin/passwd [A-Za-z]* 注：可以切换到 root 下用 passwd 命令；

```
(root) !/usr/bin/passwd root 注：可以切换到 root 下，但不能执行 passwd root 来更改 root 密码；
```

```
(root) /sbin/parted 注：可以切换到 root 下执行 parted ；
```

```
(root) /sbin/fdisk 注：可以切换到 root 下执行 fdisk ；
```

通过上面的 **sudo -l** 列出可用命令后，我想通过 **chown** 命令来改变/opt 目录的属主为 **beinan** ；

```
[beinan@localhost ~]$ ls -ld /opt 注：查看/opt 的属主；
```

```
drwxr-xr-x 26 root root 4096 10月 27 10:09 /opt 注：得到的答案是归属 root 用户和 root 用户组；
```

```
[beinan@localhost ~]$ sudo chown beinan:beinan /opt 注：通过 chown 来改变属主为 beinan 用户和 beinan 用户组；
```

```
[beinan@localhost ~]$ ls -ld /opt 注：查看/opt 属主是不是已经改变了；
```

```
drwxr-xr-x 26 beinan beinan 4096 10月 27 10:09 /opt
```

我们通过上面的例子发现 **beinan** 用户能切换到 **root** 后执行改变用户口令的 **passwd** 命令；但上面的 **sudo -l** 输出又明文写着不能更改 **root** 的口令；也就是说除了 **root** 的口令，**beinan** 用户不能更改外，其它用户的口令都能更改。下面我们来测试；

对于一个普通用户来说，除了更改自身的口令以外，他不能更改其它用户的口令。但如果换到 **root** 身份执行命令，则可以更改其它用户的口令；

比如在系统中有 **linuxsir** 这个用户，我们想尝试更改这个用户的口令，

```
[beinan@localhost ~]$ passwd linuxsir 注：不通过 sudo 直接运行 passwd 来更改 linuxsir 用户的口令；
```

```
passwd: Only root can specify a user name. 注：失败，提示仅能通过 root 来更改；
```

```
[beinan@localhost ~]$ sudo passwd linuxsir 注：我们通过/etc/sudoers 的定义，让 beinan 切换到 root 下执行 passwd 命令来改变 linuxsir 的口令；
```

```
Changing password for user linuxsir.
```

```
New UNIX password: 注：输入新口令；
```

```
Retype new UNIX password: 注：再输入一次；
```

```
passwd: all authentication tokens updated successfully. 注：改变成功；
```

后记：

本文是用户管理的文档的重要组成部份，我计划在明天开始写用户管理控制工具，比如 **useradd**、**userdel**、**usermod**，也就是管理用户的工具介绍；当然我还会写用户查询工具等与用户管理相关的

4、权限命令

4.1、chmod

文件或者目录的用户能够使用 **chmod** 命令修改文件的权限。**Chmod** 命令有两种方式：一种是字符方式，使用字符来修改文件的权限；另外一种数字方式，使用 3 个数字的组合来修改文件的权限。

使用方式：**chmod [-cfvR] [--help] [--version] mode file...**

说明：**Linux/Unix** 的档案调用权限分为三级：档案拥有者、群组、其他。利用 **chmod** 可以借以控制档案如何被他人所调用。

参数：

mode：权限设定字串，格式如下：**[ugoa...][[+|=][rwxX]...][,...]**，其中

u 表示该档案的拥有者，**g** 表示与该档案的拥有者属于同一个群体(**group**)者，**o** 表示其他以外的人，**a** 表示这三者皆是。

+ 表示增加权限、**-** 表示取消权限、**=** 表示唯一设定权限。

r 表示可读取，**w** 表示可写入，**x** 表示可执行，**X** 表示只有当该档案是个子目录或者该档案已经被设定过为可执行。

-c：若该档案权限确实已经更改，才显示其更改动作

-f: 若该档案权限无法被更改也不要显示错误讯息
-v: 显示权限变更的详细资料
-R: 对目前目录下的所有档案与子目录进行相同的权限变更(即以递归的方式逐个变更)
--help: 显示辅助说明
--version: 显示版本

范例:

将档案 file1.txt 设为所有人皆可读取: `chmod ugo+r file1.txt`

将档案 file1.txt 设为所有人皆可读取: `chmod a+r file1.txt`

将档案 file1.txt 与 file2.txt 设为该档案拥有者, 与其所属同一个群体者可写入, 但其他以外的人则不可写入: `chmod ug+w,o-w file1.txt file2.txt`

将 ex1.py 设定为只有该档案拥有者可以执行: `chmod u+x ex1.py`

将目前目录下的所有档案与子目录皆设为任何人可读取: `chmod -R a+r *`

数字方式的基本语法是: `chmod nnn 文件`

其中第 1、2、3 个 n 分别表示用户、组成员和所有其它用户。各个位置上的 n 要么是一个 0, 或者是一个由赋予权限的相关值相加得到的单个阿拉伯数字之和。这些数字的意义如表 1 所示。

值	表示的意义
4	表示文件或者目录的读权限
2	表示文件或者目录的写权限
1	表示文件或者目录的执行权限

很显然, 当使用数字方式时, 这 3 个数字必须为 0 至 7 中的一个。

若要 rwx 属性则 $4+2+1=7$;

若要 rw-属性则 $4+2=6$;

若要 r-x 属性则 $4+1=5$ 。

范例:

`chmod a=rwx file` 和 `chmod 777 file` 效果相同

`chmod ug=rwx,o=x file` 和 `chmod 771 file` 效果相同

若用 `chmod 4755 filename` 可使此程序具有 root 的权限

4.2、umask

很显然, 系统中各种文件的权限设置对特定用户的数据安全有很大影响。但是要求用户逐一明确设置系统中每个文件的权限也是不现实的, 为此, 需要使用 `umask` 命令, 该命令可以为用户账号中新文件的创建进行缺省设置。系统管理员必须要为你设置一个合理的 `umask` 值, 以确保你创建的文件具有所希望的缺省权限, 防止其他非同组用户对你的文件具有写权限。具体来说, `umask` 是用来设置权限掩码的, 权限掩码由 3 个数字组成, 将现有的存取权限减掉权限掩码后, 即可产生建立文件时默认的权限。

语法: `umask [-S][权限掩码]`

补充说明: `umask` 可用来设定[权限掩码]。[权限掩码]是由 3 个八进制的数字所组成, 将现有的存取权限减掉权限掩码后, 即可产生建立文件时预设的权限。

参数:

-S 以文字的方式来表示权限掩码。

登录之后, 可以按照个人的偏好使用 `umask` 命令来改变文件创建的缺省权限。相应的改变直到退出该 shell 或使用另外的 `umask` 命令之前一直有效。一般来说, `umask` 命令是在 `/etc/profile` 文件中设置的, 每个用户在登录时都会引用这个文件, 所以如果希望改变所有用户的 `umask`, 可以在该文件中加入相应的条目。如果希望永久性地设置自己的 `umask` 值, 那么就把它放在自己 `$HOME` 目录下的 `.profile` 或 `.bash_profile` 文件中。

如何计算 `umask` 值

`umask` 命令允许你设定文件创建时的缺省模式，对应每一类用户(文件属主、同组用户、其他用户)存在一个相应的 `umask` 值中的数字。对于文件来说，这一数字的最大值分别是 6。系统不允许你在创建一个文本文件时就赋予它执行权限，必须在创建后用 `chmod` 命令增加这一权限。目录则允许设置执行权限，这样针对目录来说，`umask` 中各个数字最大可以到 7。该命令的一般形式为：

`umask nnn`

其中 `nnn` 为 `umask` 置 000-777。

计算 `umask` 值：可以有几种计算 `umask` 值的方法，通过设置 `umask` 值，可以为新创建的文件和目录设置缺省权限。

例如，对于 `umask` 值 002，相应的文件和目录缺省创建权限是什么呢？

第一步，我们首先写下具有全部权限的模式，即 777 (所有用户都具有读、写和执行权限)。

第二步，在下面一行按照 `umask` 值写下相应的位，在本例中是 002。

第三步，在接下来的一行中记下上面两行中没有匹配的位。这就是目录的缺省创建权限。稍加练习就能够记住这种方法。

第四步，对于文件来说，在创建时不能具有文件权限，只要拿掉相应的执行权限比特即可。

这就是上面的例子，其中 `umask` 值为 002：

1) 文件的最大权限 `rw-rw-rw-` (777)

2) `umask` 值为 002 -----w-

3) 目录权限 `rw-rw-r--` (775) 这就是目录创建缺省权限

4) 文件权限 `rw-rw-r--` (664) 这就是文件创建缺省权限

系统默认的 `umask` 码是 0022 也就是：目录 755(`rw-rw-r--`)，文件：644(`rw-r--r--`)。

`umask` 码的换算

$0022 + 0755 = 0777$ 对应默认目录权限 反之 $0777 - 0755 = 0022$

$0022 + 0644 + 0111 = 0777$ 对应默认文件权限 反之 $0777 - 0111 - 0644 = 0022$

哈哈~~ 简单吧！假如我们要将默认目录权限设置为 744 那么对应的 `umask` 是 $0777 - 0744 = 0033$ ，然后执行 `umask 0033` 命令就将 `umask` 码改成 0033 了。

下面是另外一个例子，假设这次 `umask` 值为 022：

1) 文件的最大权限 `rw-rw-rw-` (777)

2) `umask` 值为 022 ---w--w-

3) 目录权限 `rw-r--r--` (755) 这就是目录创建缺省权限

4) 文件权限 `rw-r--r--` (644) 这就是文件创建缺省权限

下面是常用的 `umask` 值及对应的文件和目录权限

<code>umask</code> 值	目录	文件
022	755	644
027	750	640
002	775	664
006	771	660
007	770	660

如果想知道当前的 `umask` 值，可以使用 `umask` 命令：如果想要改变 `umask` 值，只要使用 `umask` 命令设置一个新的值即可：

`$ umask 002`

确认一下系统是否已经接受了新的 `umask` 值：在使用 `umask` 命令之前一定要弄清楚到底希望具有什么样的文件/目录创建缺省权限。否则可能会得到一些非常奇怪的结果；例如，如果将 `umask` 值设置为 600，那么所创建的文件/目录的缺省权限就是 066！除非你有特殊需要，否则没有必要去管他，系统默认的值“022”`umask` 是用的掩码，至于掩码的概念，从基础学吧，这里不说了。

linux 中的文件/目录许可是用 4 位八进制数表示的。其中第一个八进制数用来表示特殊许可设置，第二个数字用来设置文件所有者的许可，第三个数字用来设置组许可，第四个数字用来设置所有人的许可。

例如，root 的权限为 777，若权限掩码设为 022，那么两都相减后可得 755。下面是在我的系统更改 umask 的一些情况：

```
[root@linuxserver root]# umask
```

```
022
```

上述命令显示表示我的系统的 umask 值为 022。

```
[root@linuxserver root]# umask -S
```

```
u=rwx,g=rx,o=rx
```

当 umask 值为 022 时，默认情况下各用户的权限。注意这里的参数“S”是大写。

```
[root@linuxserver root]# umask 177
```

```
[root@linuxserver root]# umask -S
```

```
u=rw,g=,o=
```

上述两行命令把 umask 值改为 177，结果只有文件所有者具有读写文件的权限，其它用户不能访问该文件。这显然是一种非常安全的状态。

4.3、chown

chown 命令用途更改与文件关联的所有者或组。

语法 `chown[-f][-h][-R] Owner [:Group] { File ... | Directory ... }`

描述 chown 命令将 File 参数指定的文件的所有者更改为 Owner 参数指定的用户。Owner 参数的值可以是可在 /etc/passwd 文件中找到的用户标识或登录名。还可以选择性地指定组。Group 参数的值可以是可在 /etc/group 文件中找到的组标识或组名。

只有 root 用户可以更改文件的所有者。只在您是 root 用户或拥有该文件的情况下才可以更改文件的组。如果拥有文件但不是 root 用户，则只可以将组更改为您是其成员的组。

虽然 -H、-L 和 -P 标志是互斥的，指定不止一个也不认为是错误。指定的最后一个标志确定命令拟稿将演示的操作。

参数：

-f 禁止除用法消息之外的所有错误消息。

-h 更改遇到的符号链接的所有权，而非符号链接指向的文件或目录的所有权。当遇到符号链接而您未指定 -h 标志时，chown 命令更改链接指向的文件或目录的所有权，而非链接本身的所有权。如果指定 -R 标志，chown 命令递归地降序指定的目录。-H 如果指定了 -R 选项，并且引用类型目录的文件的符号链接在命令行上指定，chown 变量会更改由符号引用的目录的用户标识（和组标识，如果已指定）和所有在该目录下的文件层次结构中的所有文件。

-L 如果指定了 -R 选项，并且引用类型目录的文件的符号在命令行上指定或在遍历文件层次结构期间遇到，chown 命令会更改由符号链接引用的目录的用户标识（和组标识，如果已指定）和在该目录之下的文件层次结构中的所有文件。

-P 如果指定了 -R 选项并且符号链接在命令行上指定或者在遍历文件层次结构期间遇到，则如果系统支持该操作，则 chown 命令会更改符号链接的所有者标识（和组标识，如果已指定）。chown 命令不会执行至文件层次结构的任何其它部分的符号链接。

-R 递归地降序目录，更改每个文件的所有权。当遇到符号链接并且链接指向目录时，更改该目录的所有权，但不进一步遍历目录。不过 -h、-H、-L or -P 标志也未指定，则当遇到符号链接并且该链接指向到目录时，该目录的组所有权更改但不会进一步遍历目录。安全性访问控制：此程序应该作为“可信计算基”中的正常用户程序安装。退出状态该命令返回以下出口值：0 命令执行成功并已执行所有请求的更改。>0 发生错误。

示例：

要更改文件 `program.c` 的所有者： `chown jim program.c` `program.c` 的用户访问权限现在应用到 `jim`。作为所有者，`jim` 可以使用 `chmod` 命令允许或拒绝其他用户访问 `program.c`。

要将目录 `/tmp/src` 中所有文件的所有者和组更改为用户 `john` 和组 `build`： `chown -R john:build /tmp/src` 文件

将档案 `file1.txt` 的拥有者设为 `users` 群体的使用者 `jessie`：

`chown jessie:users file1.txt`

将目前目录下的所有档案与子目录的拥有者皆设为 `users` 群体的使用者 `lampport`：

`chmod -R lampport:users *`

4.4、chgrp

功能说明：变更文件或目录的所属群组。

语 法： `chgrp [-cfhRv][--help][--version][所属群组][文件或目录...]` 或 `chgrp [-cfhRv][--help][--reference=<参考文件或目录>][--version][文件或目录...]`

补充说明：在 UNIX 系统家族里，文件或目录权限的掌控以拥有者及所属群组来管理。您可以使用 `chgrp` 指令去变更文件与目录的所属群组，设置方式采用群组名称或群组识别码皆可。

参 数：

`-c` 或 `--changes` 效果类似 `-v` 参数，但仅回报更改的部分。

`-f` 或 `--quiet` 或 `--silent` 不显示错误信息。

`-h` 或 `--no-dereference` 只对符号连接的文件作修改，而不更动其他任何相关文件。

`-R` 或 `--recursive` 递归处理，将指定目录下的所有文件及子目录一并处理。

`-v` 或 `--verbose` 显示指令执行过程。

`--help` 在线帮助。

`--reference=<参考文件或目录>` 把指定文件或目录的所属群组全部设成和参考文件或目录的所属群组相同。

`--version` 显示版本信息。

范例：

```
[root@linux ~]# chgrp users install.log
```

```
[root@linux ~]# ls -l
```

```
-rw-r--r-- 1 root users 28490 Jun 25 08:53 install.log
```

```
[root@linux ~]# chgrp testing install.log
```

`chgrp:invalid group name 'testing'` <==出现错误信息～找不到这个用户组名～

发现了吗？文件的用户组被改成了 `users` 了，但要改成 `testing` 的时候，就会发生错误。注意，出现错误信息后，要检查错误信息的内容。

五、目录结构

目录结构及主要内容 “/” 根目录部分有以下子目录：

◆ `/usr` 目录包含所有的命令、程序库、文档和其它文件。这些文件在正常操作中不会被改变的。这个目录也包含你的 Linux 发行版本的主要的应用程序，譬如，`Netscape`。

◆ `/var` 目录包含在正常操作中被改变的文件：假脱机文件、记录文件、加锁文件、临时文件和页格式化文件等。这个目录中存放着那些不断在扩充着的东西，为了保持 `/usr` 的相对稳定，那些经常被修改的目录可以放在这个目录下，实际上许多系统管理员都是这样干的。顺带说一下系统的日志文件就在 `/var/log` 目录中。

- ◆/home 目录包含用户的文件：参数设置文件、个性化文件、文档、数据、EMAIL、缓存数据等。这个目录在系统省级时应该保留。
 - ◆/proc 目录整个包含虚幻的文件。它们实际上并不存在磁盘上，也不占用任何空间。（用 ls -l 可以显示它们的大小）当查看这些文件时，实际上是在访问存在内存中的信息，这些信息用于访问系统
 - ◆/bin 系统启动时需要的执行文件（二进制），这些文件可以被普通用户使用。
 - ◆/sbin 系统执行文件（二进制），这些文件不打算被普通用户使用。（普通用户仍然可以使用它们，但要指定目录。）
 - ◆/etc 操作系统的配置文件目录。
 - ◆/root 系统管理员（也叫超级用户或根用户）的 Home 目录。
 - ◆/dev 设备文件目录。LINUX 下设备被当成文件，这样一来硬件被抽象化，便于读写、网络共享以及需要临时装载到文件系统中。正常情况下，设备会有一个独立的子目录。这些设备的内容会出现在独立的子目录下。LINUX 没有所谓的驱动符。
 - ◆/lib 根文件系统目录下程序和核心模块的共享库。
 - ◆/boot 用于自举加载程序（LILO 或 GRUB）的文件。当计算机启动时（如果有多个操作系统，有可能允许你选择启动哪一个操作系统），这些文件首先被装载。这个目录也会包含 LINUX 核（压缩文件 vmlinuz），但 LINUX 核也可以存在别处，只要配置 LILO 并且 LILO 知道 LINUX 核在哪儿。
 - ◆/opt 可选的应用程序，譬如，REDHAT 5.2 下的 KDE （REDHAT 6.0 下，KDE 放在其它的 XWINDOWS 应用程序中，主执行程序在/usr/bin 目录下）
 - ◆/tmp 临时文件。该目录会被自动清理干净。
 - ◆/lost+found 在文件系统修复时恢复的文件
 - ◆/usr 目录下比较重要的部分有：
 - ◆/usr/X11R6 X-WINDOWS 系统（version 11, release 6)
 - ◆/usr/X11 同/usr/X11R6 （/usr/X11R6 的符号连接）
 - ◆/usr/X11R6/bin 大量的小 X-WINDOWS 应用程序（也可能是一些在其它子目录下大执行文件的符号连接）。
 - ◆/usr/doc LINUX 的文档资料（在更新的系统中，这个目录移到/usr/share/doc）。
 - ◆/usr/share 独立与你计算机结构的数据，譬如，字典中的词。
 - ◆/usr/bin 和/usr/sbin 类似与 “/” 根目录下对应的目录（/bin 和/sbin），但不用于基本的启动（譬如，在紧急维护中）。大多数命令在这个目录下。
 - ◆/usr/local 本地管理员安装的应用程序（也可能每个应用程序有单独的子目录）。在 “main” 安装后，这个目录可能是空的。这个目录下的内容在重安装或升级操作系统后应该存在。
 - ◆/usr/local/bin 可能是用户安装的小的应用程序，和一些在/usr/local 目录下大应用程序的符号连接。
 - ◆/proc 目录的内容：
 - ◆/proc/cpuinfo 关于处理器的信息，如类型、厂家、型号和性能等。
 - ◆/proc/devices 当前运行内核所配置的所有设备清单。
 - ◆/proc/dma 当前正在使用的 DMA 通道。/proc/filesystems 当前运行内核所配置的文件系统。
 - ◆/proc/interrupts 正在使用的中断，和曾经有多少个中断。
 - ◆/proc/ioports 当前正在使用的 I/O 端口。
- 举例，使用下面的命令能读出系统的 CPU 信息。

cat /proc/cpuinfo

/bin	bin 是 binary 的缩写。这个目录沿袭了 UNIX 系统的结构，存放着使用者最经常使用的命令。例如 cp、ls、cat，等等。
/boot	这里存放的是启动 Linux 时使用的一些核心文件。

/dev	dev 是 device（设备）的缩写。这个目录下是所有 Linux 的外部设备，其功能类似 DOS 下的 .sys 和 Win 下的 .vxd。在 Linux 中设备和文件是用同种方法访问的。例如：/dev/hda 代表第一个物理 IDE 硬盘。
/etc	这个目录用来存放系统管理所需要的配置文件和子目录。
/home	用户的主目录，比如说有个用户叫 wang，那他的主目录就是/home/wang 也可以用~wang 表示。
/lib	这个目录里存放着系统最基本的动态链接共享库，其作用类似于 Windows 里的 .dll 文件。几乎所有的应用程序都须要用到这些共享库。
/lost+found	这个目录平时是空的，当系统不正常关机后，这里就成了一些无家可归的文件的避难所。对了，有点类似于 DOS 下的 .chk 文件。
/mnt	这个目录是空的，系统提供这个目录是让用户临时挂载别的文件系统。
/proc	这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。也就是说，这个目录的内容不在硬盘上而是在内存里。
/root	系统管理员（也叫超级用户）的主目录。作为系统的拥有者，总要有些特权啊！比如单独拥有一个目录。
/sbin	s 就是 Super User 的意思，也就是说这里存放的是系统管理员使用的管理程序。
/tmp	这个目录不用说，一定是用来存放一些临时文件的地方了。
/usr	这是最庞大的目录，我们要用到的应用程序和文件几乎都存放在这个目录下。其中包含以下子目录：
/usr/X11R6	存放 X-Window 的目录；
/usr/bin	存放着许多应用程序；
/usr/sbin	给超级用户使用的一些管理程序就放在这里；
/usr/doc	这是 Linux 文档的大本营；
/usr/include	Linux 下开发和编译应用程序需要的头文件，在这里查找；
/usr/lib	存放一些常用的动态链接共享库和静态档案库；
/usr/local	这是提供给一般用户的/usr 目录，在这里安装软件最适合；
/usr/man	man 在 Linux 中是帮助的同义词，这里就是帮助文档的存放目录；
/usr/src	Linux 开放的源代码就存在这个目录，爱好者们别放过哦！
/var	这个目录中存放着那些不断在扩充着的东西，为了保持/usr 的相对稳定，那些经常被修改的目录可以放在这个目录下，实际上许多系统管理员都是这样干的。顺带说一下系统的日志文件就在/var/log 目录中。

总结来说：

- ◆用户应该将文件存在/home/user_login_name 目录下(及其子目录下)。
- ◆本地管理员大多数情况下将额外的软件安装在/usr/local 目录下并符号连接在/usr/local/bin 下的主执行程序。
- ◆系统的所有设置在/etc 目录下。
- ◆不要修改根目录（“/”）或/usr 目录下的任何内容，除非真的清楚要做什么。这些目录最好和 LINUX 发布时保持一致。

- ◆大多数工具和应用程序安装在目录：`/bin, /usr/sbin, /sbin, /usr/x11/bin, /usr/local/bin`。
- ◆所有的文件在单一的目录树下。没有所谓的“驱动符”。

六、软件安装

RPM

RPM 软件的安装、删除、更新只有 root 权限才能使用；对于查询功能任何用户都可以操作；如果普通用户拥有安装目录的权限，也可以进行安装。

初始化 rpm 数据库

通过 rpm 命令查询一个 rpm 包是否安装了，也是要通过 rpm 数据库来完成的；所以我们要经常用下面的两个命令来初始化 rpm 数据库：

```
[root@localhost beinan]# rpm --initdb
[root@localhost beinan]# rpm --rebuilddb 注：这个要花好长时间；
```

注：这两个参数是极为有用，有时 rpm 系统出了问题，不能安装和查询，大多是这里出了问题；
`/var/lib/rpm` 目录下的数据库记录所有软件的升级需求，记录已经安装的所有软件，数字证书记录等，这个目录下的文件非常重要。

RPM软件包管理的查询功能：

命令格式

```
rpm {-q|--query} [select-options] [query-options]
```

RPM 的查询功能是极为强大，是极为重要的功能之一；举几个常用的例子，更为详细的具体的，请参考
`#man rpm`

1、查询系统已安装的软件；

语法：`rpm -q 软件名`

举例：

```
[root@localhost beinan]# rpm -q gaim
gaim-1.3.0-1.fc4
```

`-q` 就是 `--query`，中文意思是“问”，此命令表示的是，是不是系统安装了 gaim；如果已安装会有信息输出；如果没有安装，会输出 gaim 没有安装的信息；查看系统中所有已经安装的包，要加 `-a` 参数；

```
[root@localhost RPMS]# rpm -qa
```

如果分页查看，再加一个管道 `|` 和 `more` 命令；

```
[root@localhost RPMS]# rpm -qa |more
```

在所有已经安装的软件包中查找某个软件，比如说 gaim；可以用 `grep` 抽取出来；

```
[root@localhost RPMS]# rpm -qa |grep gaim
```

上面这条的功能和 `rpm -q gaim` 输出的结果是一样的；
等 37448

2、查询一个已经安装的文件属于哪个软件包；

语法 `rpm -qf 文件名` 注：文件名所在的绝对路径要指出举例：

```
[root@localhost RPMS]# rpm -qf /usr/lib/libacl.la
libacl-devel-2.2.23-8
```

3、查询已安装软件包都安装到何处；

语法：`rpm -ql 软件名` 或 `rpm rpmquery -ql 软件名`

举例：

```
[root@localhost RPMS]# rpm -ql lynx
[root@localhost RPMS]# rpmquery -ql lynx
```

4、查询一个已安装软件包的信息

语法格式： rpm -qi 软件名

举例：

```
[root@localhost RPMS]# rpm -qi lynx
```

5、查看一下已安装软件的配置文件；

语法格式： rpm -qc 软件名

举例：

```
[root@localhost RPMS]# rpm -qc lynx
```

6、查看一个已经安装软件的文档安装位置：

语法格式： rpm -qd 软件名

举例：

```
[root@localhost RPMS]# rpm -qd lynx
```

7、查看一下已安装软件所依赖的软件包及文件；

语法格式： rpm -qR 软件名

举例：

```
[root@localhost beinan]# rpm -qR rpm-python
```

查询已安装软件的总结：对于一个软件包已经安装，我们可以把一系列的参数组合起来用；比如 rpm -qil ；比如：

```
[root@localhost RPMS]# rpm -qil lynx
```

对已安装软件包查询的一点补充；

```
[root@localhost RPMS]# updatedb
```

```
[root@localhost RPMS]# locate 软件名或文件名
```

通过 updatedb,我们可以用 locate 来查询一些软件安装到哪里了；系统初次安装时要执行 updatedb , 每隔一段时间也要执行一次；以保持已安装软件库最新；updatedb 是 slocate 软件包所有；如果您没有这个命令，就得安装 slocate ；举例：

```
[root@localhost RPMS]# locate gaim
```

对于未安装的软件包的查看：

查看的前提是您有一个.rpm 的文件，也就是说对既有软件 file.rpm 的查看等；

1、查看一个软件包的用途、版本等信息；

语法： rpm -qpi file.rpm

举例：

```
[root@localhost RPMS]# rpm -qpi lynx-2.8.5-23.i386.rpm
```

2、查看一件软件包所包含的文件；

语法： rpm -qpl file.rpm

举例：

```
[root@localhost RPMS]# rpm -qpl lynx-2.8.5-23.i386.rpm
```

3、查看软件包的文档所在的位置；

```
语法： rpm -qpd file.rpm
```

举例：

```
[root@localhost RPMS]# rpm -qpd lynx-2.8.5-23.i386.rpm
```

4、查看一个软件包的配置文件；

```
语法： rpm -qpc file.rpm
```

举例：

```
[root@localhost RPMS]# rpm -qpc lynx-2.8.5-23.i386.rpm
```

5、查看一个软件包的依赖关系

```
语法： rpm -qpR file.rpm
```

举例：

```
[root@localhost archives]# rpm -qpR yumex_0.42-3.0.fc4_noarch.rpm
/bin/bash
/usr/bin/python
config(yumex) = 0.42-3.0.fc4
pygtk2
pygtk2-libglade
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
usermode
yum >= 2.3.2
```

软件包的安装、升级、删除等；

1、安装和升级一个rpm 包；

```
[root@localhost beinan]#rpm -vih file.rpm 注：这个是用来安装一个新的 rpm 包；
```

```
[root@localhost beinan]#rpm -Uvh file.rpm 注：这是用来升级一个 rpm 包；
```

如果有依赖关系的，请解决依赖关系，其实软件包管理器能很好的解决依赖关系，请看前面的软件包管理器的介绍；如果您在软件包管理器中也找不到依赖关系的包；那只能通过编译他所依赖的包来解决依赖关系，或者强制安装；语法结构：

```
[root@localhost beinan]# rpm -ivh file.rpm --nodeps --force
```

```
[root@localhost beinan]# rpm -Uvh file.rpm --nodeps --force
```

更多的参数，请查看 man rpm 举例应用：

```
[root@localhost RPMS]# rpm -ivh lynx-2.8.5-23.i386.rpm
```

```
Preparing... ##### [100%]
```

```
1:lynx ##### [100%]
```

```
[root@localhost RPMS]# rpm -ivh --replacepkgs lynx-2.8.5-23.i386.rpm
```

```
Preparing... ##### [100%]
```

```
1:lynx ##### [100%]
```

注： --replacepkgs 参数是以已安装的软件再安装一次；有时没有太大的必要；测试安装参数 --test ，用来检查依赖关系；并不是真正的安装；

```
[root@localhost RPMS]# rpm -ivh --test gaim-1.3.0-1.fc4.i386.rpm
```

```
Preparing... ##### [100%]
```

由新版本降级为旧版本，要加 `--oldpackage` 参数；

```
[root@localhost RPMS]# rpm -qa gaim
gaim-1.5.0-1.fc4
[root@localhost RPMS]# rpm -Uvh --oldpackage gaim-1.3.0-1.fc4.i386.rpm
Preparing... ##### [100%]
 1:gaim ##### [100%]
[root@localhost RPMS]# rpm -qa gaim
gaim-1.3.0-1.fc4
```

为软件包指定安装目录：要加 `-relocate` 参数；下面的举例是把 `gaim-1.3.0-1.fc4.i386.rpm` 指定安装在 `/opt/gaim` 目录中；

```
[root@localhost RPMS]# rpm -ivh --relocate /=/opt/gaim gaim-1.3.0-1.fc4.i386.rpm
Preparing... ##### [100%]
 1:gaim ##### [100%]
[root@localhost RPMS]# ls /opt/
gaim
```

为软件包指定安装目录：要加 `-relocate` 参数；下面的举例是把 `lynx-2.8.5-23.i386.rpm` 指定安装在 `/opt/lynx` 目录中；

```
[root@localhost RPMS]# rpm -ivh --relocate /=/opt/lynx --badreloc lynx-2.8.5-23.i386.rpm
Preparing... ##### [100%]
1:lynx ##### [100%]
```

我们安装在指定目录中的程序如何调用呢？一般执行程序，都放在安装目录的 `bin` 或者 `sbin` 目录中；看下面的例子；如果有错误输出，就做相应的链接，用 `ln -s`；

```
[root@localhost RPMS]# /opt/lynx/usr/bin/lynx
Configuration file /etc/lynx.cfg is not available.
[root@localhost RPMS]# ln -s /opt/lynx/etc/lynx.cfg /etc/lynx.cfg
[root@localhost RPMS]# /opt/lynx/usr/bin/lynx www.linuxsir.org
```

RPM 管理包管理器支持网络安装和查询；

比如我们想通过 Fedora Core 4.0 的一个镜像查询、安装软件包；地址：

<http://mirrors.kernel.org/fedora/core/4/i386/os/Fedora/RPMS/> 举例：命令格式：

```
rpm 参数 rpm 包文件的 http 或者 ftp 的地址
# rpm -qp http://mirrors.kernel.org/fedora/core/4/i386/os/
Fedora/RPMS/gaim-1.3.0-1.fc4.i386.rpm
# rpm -ivh http://mirrors.kernel.org/fedora/core/4/i386/os/
Fedora/RPMS/gaim-1.3.0-1.fc4.i386.rpm
```

2、删除一个rpm 包；

首先您要学会查询 rpm 包；请看前面的说明； `[root@localhost beinan]#rpm -e 软件包名` 举例：我想移除 `lynx` 包，完整的操作应该是：

```
[root@localhost RPMS]# rpm -e lynx
```

如果有依赖关系，您也可以用 `--nodeps` 忽略依赖的检查来删除。但尽可能不要这么做，最好用软件包管理器 `system-config-packages` 来删除或者添加软件；

```
[root@localhost beinan]# rpm -e lynx --nodeps
```

RPM验证与数字证书：

导入签名：


```
[root@localhost RPMS]# rpm --import 签名文件 举例:
```

```
[root@localhost fc40]# rpm --import RPM-GPG-KEY
```

```
[root@localhost fc40]# rpm --import RPM-GPG-KEY-fedora
```

RPM 验证作用是使用 /var/lib/rpm 下面的数据库内容来比较目前 linux 系统的环境下的所有软件文件, 也就是说当你有数据不小心丢失, 或者不小心修改到某个软件的文件内容, 就用这个简单的方法验证一下原本的文件系统

```
#rpm -Va          列出目前系统上面所有可能被改动过的文件
```

从rpm软件包抽取文件;

命令格式: `rpm2cpio file.rpm |cpio -div`

举例:

```
[root@localhost RPMS]# rpm2cpio gaim-1.3.0-1.fc4.i386.rpm |cpio -div
```

抽取出来的文件就在当前操作目录中的 `usr` 和 `etc` 中; 其实这样抽到文件不如指定安装目录来安装软件来的方便; 也一样可以抽出文件; 为软件包指定安装目录: 要加 `-relocate` 参数; 下面的举例是把 `gaim-1.3.0-1.fc4.i386.rpm` 指定安装在 `/opt/gaim` 目录中;

```
[root@localhost RPMS]# rpm -ivh --relocate /=/opt/gaim gaim-1.3.0-1.fc4.i386.rpm
```

```
Preparing... ##### [100%]
```

```
1:gaim ##### [100%]
```

```
[root@localhost RPMS]# ls /opt/
```

```
gaim
```

这样也能一目了然; `gaim` 的所有文件都是安装在 `/opt/gaim` 中, 我们只是把 `gaim` 目录备份一下, 然后卸掉 `gaim`; 这样其实也算提取文件的一点用法;

RPM的配置文件;

RPM 包管理, 的配置文件是 `rpmrc`, 我们可以在自己的系统中找到; 比如 Fedora Core 4.0 中的 `rpmrc` 文件位于;

```
[root@localhost RPMS]# locate rpmrc
```

```
/usr/lib/rpm/rpmrc
```

```
/usr/lib/rpm/redhat/rpmrc
```

我们可以通过 `rpm --showrc` 查看; 具体的还得我们自己来学习。呵。。。不要问我, 我也不懂; 只要您看了这篇文章, 认为对您有用, 您的水平就和我差不多; 咱们水平是一样的, 所以我不能帮助您了; 请理解。

YUM

YUM配置文件

创建容器, 位置在 `/etc/yum.repos.d`, 扩展名必须是 `.repo`

```
#cd /etc/yum.repos.d
```

```
#vim yum.repo    新建一个仓库文件, 名字可以随便定义, 在文件中写如下内容
```

```
[base]          #代表容器名称, 中括号一定要存在, 里面的名字可随便取
```

```
name=base       #说明这个容器的意义, 随便写都可以
```

```
baseurl=ftp://192.168.0.6/pub/Server    #192.168.0.6 是你的 YUM 源地址, 这个很重要。
```

```
enabled=1       #是否启动, =0 则不启动, 不启动就无法使用该源
```

```
gpgcheck=0      #是否验证. 可不要
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release    #验证的密钥. 可不要
```

命令:#yum repolist all 显示目前所使用的容器有哪些

如果查询出来的容器，status 为 disabled，要将配置文件，如上 enabled=1

/etc/yum.conf

yum.conf 这个配置文件主要是 **yum** 客户端使用，里面主要规定了要去用的 **rpm** 包的 **yum** 服务器的信息。

[main] #main 开头的块用于对客户端进行配置，在 main 后也可以指定 yum 源（不推荐这样做），与 /etc/yum.repo.d 中指定 yum 源相同

cachedir=/var/cache/yum

#cachedir: yum 缓存的目录，yum 在此存储下载的 rpm 包和数据库，一般是/var/cache/yum。

keepcache=0 #0 表示不保存下载的文件，1 表示保存下载的文件，默认为不保存

debuglevel=2

#debuglevel: 除错级别，0——10,默认是 2 貌似只记录安装和删除记录

logfile=/var/log/yum.log #指定 yum 的日志文件

pkgpolicy=newest #包的策略，如果配置多了 yum 源，同一软件在不同的 yum 源中有不同版本，newest 则安装最新版本，该值为 lastest，则 yum 会将服务器上 ID 按照字母序排列，选择最后那个服务器上的软件安装

distroverpkg=centos-release

#指定一个软件包，yum 会根据这个包判断你的发行版本，默认是 redhat-release，也可以是安装的任何针对自己发行版的 rpm 包。

tolerant=1

#tolerant, 也有 1 和 0 两个选项，表示 yum 是否容忍命令行发生与软件包有关的错误，比如你要安装 1,2,3 三个包，而其中 3 此前已经安装了，如果你设为 1,则 yum 不会出现错误信息。默认是 0。

exactarch=1

#exactarch, 有两个选项 1 和 0,代表是否只升级和你安装软件包 cpu 体系一致的包，如果设为 1，则如你安装了一个 i386 的 rpm，则 yum 不会用 i686 的包来升级。

retries=20

#retries, 网络连接发生错误后的重试次数，如果设为 0，则会无限重试。

obsoletes=1

gpgcheck=1

#gpgcheck= 有 1 和 0 两个选择，分别代表是否是否进行 gpg 校验，如果没有这一项，默认是检查的。

plugins = 1 #是否启用插件，默认 1 为允许，0 表示不允许

reposdir=/etc/yy.rm #默认是 /etc/yum.repos.d/ 低下的 xx.repo 后缀文件

#默认都会被 include 进来 也就是说 /etc/yum.repos.d/xx.repo 无论配置文件有多少个 每个里面有多少个[name]最后其实都被整合到 一个里面看就是了 重复的[name]应该是前面覆盖后面的--还是后面的覆盖前面的呢? enabled 测试是后面覆盖前面

exclude=xxx

#exclude 排除某些软件在升级名单之外，可以用通配符，列表中各个项目要用空格隔开，这个对于安装了诸如美化包，中文补丁的朋友特别有用。

keepcache=[1 or 0]

#设置 keepcache=1, yum 在成功安装软件包之后保留缓存的头文件 (headers) 和软件包。默认值为 keepcache=0 不保存

reposdir=[包含 .repo 文件的目录的绝对路径]

#该选项用户指定 .repo 文件的绝对路径。.repo 文件包含软件仓库的信息 (作用与 /etc/yum.conf 文件中的 [repository] 片段相同)。中

YUM命令

用 YUM 安装删除软件 `yum install xxx`, yum 会查询数据库, 有无这一软件包, 如果有, 则检查其依赖冲突关系, 如果没有依赖冲突, 那么最好, 下载安装;如果有, 则会给出提示, 询问是否要同时安装依赖, 或删除冲突的包, 你可以自己作出判断。

删除的命令是, `yum remove xxx`, 同安装一样, yum 也会查询数据库, 给出解决依赖关系的提示。

YUM安装软件包

命令: `yum install`

YUM删除软件包

命令: `yum remove`

用 YUM 查询软件信息, 我们常会碰到这样的情况, 想要安装一个软件, 只知道它和某方面有关, 但又不能确切知道它的名字。这时 yum 的查询功能就起作用了。你可以用 `yum search keyword` 这样的命令来进行搜索, 比如我们要安装一个 Instant Messenger, 但又不知到底有哪些, 这时不妨用 `yum search messenger` 这样的指令进行搜索, yum 会搜索所有可用 rpm 的描述, 列出所有描述中和 messenger 有关的 rpm 包, 于是我们可能得到 gaim, kopete 等等, 并从中选择。有时我们还会碰到安装了一个包, 但又不知道其用途, 我们可以用 `yum info packagename` 这个指令来获取信息。

1.使用 YUM 查找软件包

命令: `yum search`

2.列出所有可安装的软件包

命令: `yum list`

3.列出所有可更新的软件包

命令: `yum list updates`

4.列出所有已安装的软件包

命令: `yum list installed`

5.列出所有已安装但不在 Yum Repository 内的软件包

命令: `yum list extras`

6.列出所指定的软件包

命令: `yum list`

7.使用 YUM 获取软件包信息

命令: `yum info`

8.列出所有软件包的信息

命令: `yum info`

9.列出所有可更新的软件包信息

命令: `yum info updates`

10.列出所有已安装的软件包信息

命令: `yum info installed`

11.列出所有已安装但不在 Yum Repository 内的软件包信息

命令: `yum info extras`

12.列出软件包提供哪些文件

命令: `yum provides`

清除YUM缓存

yum 会把下载的软件包和 header 存储在 cache 中，而不会自动删除。如果我们觉得它们占用了磁盘空间，可以使用 yum clean 指令进行清除，更精确的用法是 yum clean headers 清除 header，yum clean packages 清除下载的 rpm 包，yum clean all 一股脑儿端

1.清除缓存目录(/var/cache/yum)下的软件包

命令：yum clean packages

2.清除缓存目录(/var/cache/yum)下的 headers

命令：yum clean headers

3.清除缓存目录(/var/cache/yum)下旧的 headers

命令：yum clean

Oldheaders

4.清除缓存目录(/var/cache/yum)下的软件包及旧的 headers

命令：yum clean, yum clean

all (= yum clean packages; yum clean oldheaders)

七、时间管理

1、Linux时间介绍：

Linux 时钟分为系统时钟（System Clock）和硬件（Real Time Clock，简称 RTC）时钟。系统时钟是指当前 Linux Kernel 中的时钟，而硬件时钟则是主板上由电池供电的时钟，这个硬件时钟可以在 BIOS 中进行设置。当 Linux 启动时，硬件时钟会去读取系统时钟的设置，然后系统时钟就会独立于硬件运作。

Linux 中的所有命令（包括函数）都是采用的系统时钟设置。在 Linux 中，用于时钟查看和设置的命令主要有 date、hwclock 和 clock。其中，clock 和 hwclock 用法相近，只用一个就行，只不过 clock 命令除了支持 x86 硬件体系外，还支持 Alpha 硬件体系。

2、Linux时间设置命令

2.1、date:

语法格式：date [-u] [-d datestr] [-s datestr] [--utc] [--universal] [--date=datestr] [--set=datestr] [--help] [--version] [+FORMAT] [MMDDhhmm[[CC]YY][.ss]]

说明：可用来设置系统日期与时间。只有管理员才有设置日期与时间的权限，一般用户只能用 date 命令显示时间。若不加任何参数，date 会显示目前的日期与时间。

例 1：显示当前系统时间

```
[root@Test2 ~]# date
```

2010 年 06 月 17 日 星期四 00:00:04 CST

例 2：设置日期和时间 2010 年 6 月 18 号 12:00

```
[root@Test2 ~]# date -s "20100618 12:00:00"
```

2010 年 06 月 18 日 星期五 12:00:00 CST

例 3：设置日期为 2010 年年 6 月 18 号

```
[root@Test2 ~]# date -s 20100618
```

2010 年 06 月 18 日 星期五 00:00:00 CST

例 4：设置时间为 12:00:00

```
[root@Test2 ~]# date 12:00:00
date: invalid date "12:00:00"
```

例 5：显示时区

```
[root@Test2 ~]# date -R
Thu, 17 Jun 2010 00:01:36 +0800
```

或者：

```
[root@Test2 ~]# cat /etc/sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
# The timezone of the system is defined by the contents of /etc/localtime.
ZONE="Asia/Shanghai"
UTC=true
ARC=false
```

2.2、hwclock/clock

语法格式：hwclock [--adjust][--debug][--directisa][--hctosys][--show][--systohc][--test]
[--utc][--version][--set --date=<日期与时间>]

参数：

--adjust hwclock 每次更改硬件时钟时，都会记录在/etc/adjtime 文件中。使用--adjust 参数，可使 hwclock 根据先前的记录来估算硬件时钟的偏差，并用来校正目前的硬件时钟。

--debug 显示 hwclock 执行时详细的信息。

--directisa hwclock 预设从/dev/rtc 设备来存取硬件时钟。若无法存取时，可用此参数直接以 I/O 指令来存取硬件时钟。

--hctosys 将系统时钟调整为与目前的硬件时钟一致。

--set --date=<日期与时间> 设定硬件时钟。

--show 显示硬件时钟的时间与日期。

--systohc 将硬件时钟调整为与目前的系统时钟一致。

--test 仅测试程序，而不会实际更改硬件时钟。

--utc 若要使用格林威治时间，请加入此参数，hwclock 会执行转换的工作。

--version 显示版本信息。

例 1：查看硬件时间

```
# hwclock --show
```

或者

```
# clock --show
```

例 2：设置硬件时间

```
# hwclock --set --date="07/07/06 10:19" （月/日/年 时:分:秒）
```

或者

```
# clock --set --date="07/07/06 10:19" （月/日/年 时:分:秒）
```

例 3：硬件时间和系统时间的同步

按照前面的说法，重新启动系统，硬件时间会读取系统时间，实现同步，但是在不重新启动的时候，需要用 hwclock 或 clock 命令实现同步。

硬件时钟与系统时钟同步：

```
# hwclock --hctosys （hc 代表硬件时间，sys 代表系统时间）
```

或者

```
# clock -hctosys
```

例 4：系统时钟和硬件时钟同步：

```
# hwclock --systohc
```

或者

```
# clock --systohc
```

例 5：强制将系统时间写入 CMOS，使之永久生效，避免系统重启后恢复成原时间

```
# clock -w
```

或者

```
# hwclock -w
```

2.3、时区的设置

```
# 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.

#? 输入 5，亚洲

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

#? 输入 9，中国

Please select one of the following time zone regions.

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

- 2) Heilongjiang
- 3) central China - Gansu, Guizhou, Sichuan, Yunnan, etc.
- 4) Tibet & most of Xinjiang Uyghur
- 5) southwest Xinjiang Uyghur

#? 输入 1, 北京时间

The following information has been given:

China

east China - Beijing, Guangdong, Shanghai, etc.

Therefore TZ='Asia/Shanghai' will be used.

Local time is now: Fri Jul 7 10:32:18 CST 2006.

Universal Time is now: Fri Jul 7 02:32:18 UTC 2006.

Is the above information OK?

1) Yes

2) No

#? 输入 1, 确认

如果不用 tzselect 命令, 可以修改文件变更时区。

vi /etc/sysconfig/clock

Z/Shanghai (查/usr/share/zoneinfo 下面的文件)

UTC=false

ARC=false

rm /etc/localtime

ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

重新启动即可。

2.4、图形界面设置时区命令timeconfig

2.5、时间同步

例 1: 同步时间

ntpdate 210.72.145.44 (210.72.145.44 是中国国家授时中心的官方服务器)

例 2: 定时同步时间

crontab -e 添加脚本例子如下:

*/20 * * * * /usr/sbin/ntpdate 210.72.145.44 //每 20 分钟执行一次

30 5 * * * /usr/sbin/ntpdate 210.72.145.44 //每天早晨 5 点半执行

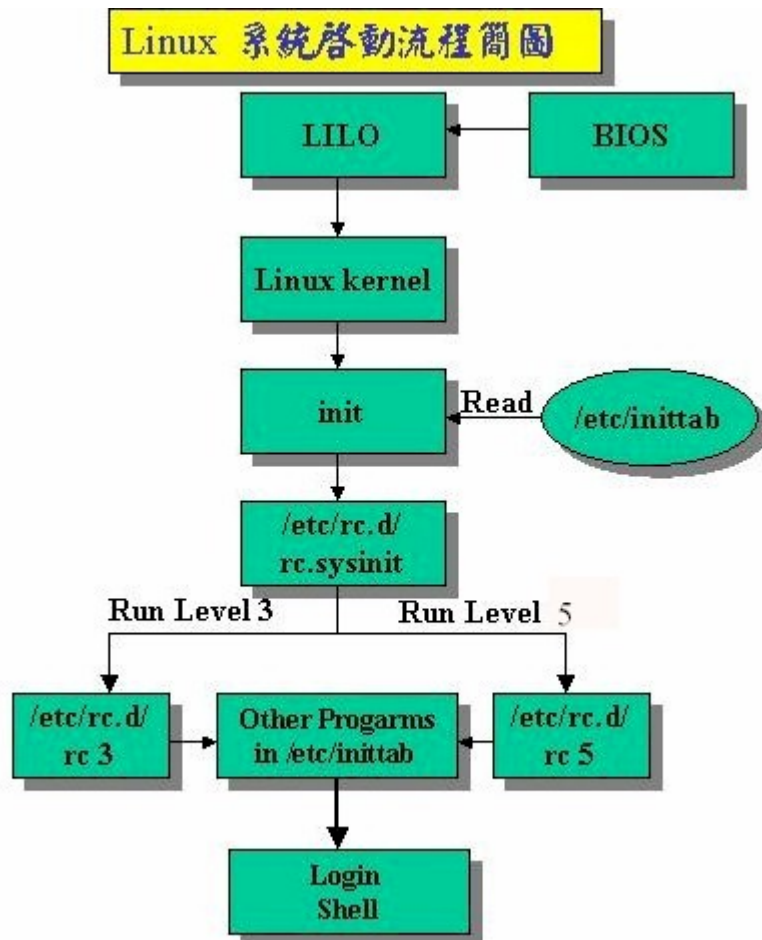
※ 前面五个*号代表五个数字, 数字的取值范围和含义如下: 分钟(0-59) 小时(0-23) 日期(1-31) 月份(1-12) 星期(0-6) //0 代表星期天设定完毕后, 可使用# crontab -l 查看上面的设定。

八、启动引导

1、Linux的启动流程

- 1) BIOS 自检
- 2) 启动 GRUB/LILO
- 3) 运行 Linux kernel 并检测硬件
- 4) 挂载根文件系统
- 5) 运行 Linux 系统的第一个进程 init(其 PID 永远为 1, 是所有其它进程的父进程)
- 6) init 读取系统引导配置文件 /etc/inittab 中的信息进行初始化

- 7) 执行系统初始化脚本— /etc/rc.d/rc.sysinit, 执行系统初始化(包括很多内容)
 - 8) 根据指定的运行级别(runlevel)来运行服务器脚本程序, 再执行脚本 /etc/rc.d/rc.local
 - 9) 运行一些其他的特别服务, 一般为 /sbin/mingetty 和 /etc/X11/prefdm
 - 10) Linux 控制台(console)提示用户输入用户名、密码进行登陆。
- 总结: BIOS 初始化→检查外围设备→检查启动设备→读区 MBR



2、在Linux中常用的启动引导工具: grub和lilo

在 Linux 和 WINDOWS 两系统并存时就需要安装 GRUB (Grand Unified Bootloader), GRUB 被广泛地用于替代 lilo, GRUB 支持在启动时使用命令行模式, 支持 md5 加密保护 还可以从 ext2/ext3、ReiseFS、JFS、FAT、minix 及 FFS 文件系统上启动其配置文件为 /boot/grub/grub.conf, 更改 grub.conf 即可立时生效如果硬盘上的 MBR 被更动过, 可以用 /sbin/grub-install /dev/hda 来重安装 grub 现在我们打开 /boot/grub/grub.conf 查看一下

```
# vim /boot/grub/grub.conf
```

内容如下:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE: You have a /boot partition. This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/sda2
```


initrd /initrd-version.img

#boot=/dev/sda

default=0

#default=0 表示默认启动第一个系统,如果系统有两个系统是用什么做为分隔符的呢? title 就是系统的分隔符,第一个 title 后面就是第一个系统,用 0 表示。

timeout=5

#timeout=5, 就是默认在启动选择界面停留的时间, 单位是秒。等待 5 秒自动进入默认操作系统

splashimage=(hd0,0)/grub/splash.xpm.gz

#splashimage 是 grub 启动背景画面, 如果是自己写 grub.conf 文件, 这个可以不用写。

hiddenmenu

title Red Hat Enterprise Linux Server (2.6.18-53.el5)

#title 后面就是系统在启动时候显示的名字

root (hd0,0)

#root 启动文件所在位置

kernel /vmlinuz-2.6.18-53.el5 ro root=LABEL=/ rhgb quiet

#kernel 内核所在位置和名字

initrd /initrd-2.6.18-53.el5.img

#initrd 内核镜像的名字

grub.conf 的范例:

timeout=10 #等待 10 秒自动进入默认操作系统

splashimage=(hd0,0)/grub/splash.xpm.gz #grub 启动背景画面

default=0 #默认进入第一个标题

title Red Hat Linux (2.4.20-18) #Red Hat Linux 标题

root (hd0,0) #根文件系统位置

kernel /vmlinuz-2.4.20-18 ro root=LABEL=/ #核心位置与核心加载参数

initrd /initrd-2.4.20-18.img #启动 initrd ram 盘

title windows #另一个操作系统的标题

rootnoverify (hd0,1) #操作系统存放在 hd0,1 上, 不要在 grub 里 mount

chainloader +1 #从 hd0,1 的第一个扇面启动

九、运行级别

1、Linux 系统的运行级别(runlevel)

Linux 系统有 7 个运行级别, Linux 系统任何时候都运行在一个指定的运行级别上, 不同的运行级别所运行的程序和服务不尽相同, 所要完成的工作和要达到的目的也不相同

- 运行级别 0 系统停机(halt)状态, 系统的默认运行级别不能设为 0, 否则不能正常启动
- 运行级别 1 单用户工作(single user)状态, root 权限, 用于系统维护, 禁止远程登陆
- 运行级别 2 多用户(multiuser)状态 (没有 NFS)
- 运行级别 3 完全的多用户(multiuser)状态 (有 NFS), 登陆后进入控制台命令行模式
- 运行级别 4 系统未使用, 保留
- 运行级别 5 X11 控制台 (xdm、gdm、kdm), 登陆后进入图形 GUI 模式

- 运行级别 6 系统正常关闭并重启(reboot)，默认运行级别不能设为 6，否则不能正常启动

2、运行级别的原理

在目录 `/etc/rc.d/init.d` 下有许多服务器脚本程序，一般称为服务(service)，在 `/etc/rc.d` 下有 7 个名为 `rcN.d` 的目录，其中 `N=0-6`，对应于系统的 7 个运行级别，`rcN.d` 目录下，都是一些符号链接文件，这些链接文件都指向 `init.d` 目录下的 `service` 脚本文件，这些链接文件的命名规则是 "`K+nn+服务名`" 或 "`S+nn+服务名`"，其中 `nn` 为 2 位数字：

例：`rc3.d` 目录下的链接文件 `S80sendmail` 就指向 `service` 脚本文件 `../init.d/sendmail`

系统会根据指定的 `runlevel` 进入对应的 `rcN.d` 目录，并按照文件名顺序检索目录下的链接文件

- 对于以 `K` 为开头的链接文件，系统将终止对应的服务
- 对于以 `S` 为开头的链接文件，系统将启动对应的服务

通过这种方式来实现 "不同的运行级别运行不同的程序和服务"

3、`/etc/inittab`配置文件详解

`init` 的进程号是 1，从这一点就能看出，`init` 进程是系统所有进程的起点，Linux 在完成内核引导以后，就开始运行 `init` 程序，`init` 程序需要读取设置文件 `/etc/inittab`。`inittab` 是个不可执行的文本文件，他有若干行指令所组成。在 Redhat 系统中，`inittab` 的内容如下所示(以 "`###`" 开始的中注释为笔者增加的)：

(如果你改变了 `inittab` 文件，那么要使他立即生效，需要使用一个命令：`init q`)

```
#
# inittab          This file describes how the INIT process should set up
#                  the system in a certain run-level.
#
# Author:          Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#                  Modified for RHS Linux by Marc Ewing and Donnie Barnes
#
# Default runlevel. The runlevels used by RHS are:
#  0 - halt (Do NOT set initdefault to this)
#  1 - Single user mode
#  2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#  3 - Full multiuser mode
#  4 - unused
#  5 - X11
#  6 - reboot (Do NOT set initdefault to this)
#
```

###表示当前缺省运行级别为 5(initdefault);

id:5:initdefault:

###启动时自动执行/etc/rc.d/rc.sysinit 脚本(sysinit)

System initialization.

si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0

l1:1:wait:/etc/rc.d/rc 1

l2:2:wait:/etc/rc.d/rc 2

```
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
###当运行级别为 5 时，以 5 为参数运行/etc/rc.d/rc 脚本，init 将等待其返回(wait)
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

###在启动过程中允许按 CTRL-ALT-DELETE 重启系统

Trap CTRL-ALT-DELETE

ca::ctrlaltdel:/sbin/shutdown -t3 -r now

When our UPS tells us power has failed, assume we have a few minutes

of power left. Schedule a shutdown for 2 minutes from now.

This does, of course, assume you have powerd installed and your

UPS connected and working correctly.

pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

If power was restored before the shutdown kicked in, cancel it.

pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

###在 2、3、4、5 级别上以 ttyX 为参数执行/sbin/mingetty 程序，打开 ttyX 终端用于用户登录，

###如果进程退出则再次运行 mingetty 程序(respawn)

Run gettys in standard runlevels

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

4:2345:respawn:/sbin/mingetty tty4

5:2345:respawn:/sbin/mingetty tty5

6:2345:respawn:/sbin/mingetty tty6

###在 5 级别上运行 xdm 程序，提供 xdm 图像方式登录界面，并在退出时重新执行(respawn)

Run xdm in runlevel 5

x:5:respawn:/etc/X11/prefdm -nodaemon

以上的 inittab 文件为例，来说明一下 inittab 的格式。其中以#开始的行是注释行，除了注释行之外，每一行都有以下格式：

id:runlevel:action:process

对上面各项的周详解释如下：

1. id

id 是指入口标识符，他是个字符串，对于 getty 或 mingetty 等其他 login 程序项，需求 id 和 tty 的编号相同，否则 getty 程序将不能正常工作。

2. runlevel

runlevel 是 init 所处于的运行级别的标识，一般使用 0—6 及 S 或 s。0、1、6 运行级别被系统保留：其中 0 作为 shutdown 动作，1 作为重启至单用户模式，6 为重启；S 和 s 意义相同，表示单用户模式，且无需 inittab 文件，因此也不在 inittab 中出现，实际上，进入单用户模式时，init 直接在控制台（/dev/console）上运行/sbin/sulogin。在一般的系统实现中，都使用了 2、3、4、5 几个级别，在 Redhat 系统中，2 表示无 NFS 支持的多用户模式，3 表示完全多用户模式（也是最常用的级别），4 保留给用户自定义，5 表示 XDM 图像登录方式。7—9 级别也是能使用的，传统的 Unix 系统没有定义这几个级别。runlevel 能是并列的多个值，以匹配多个运行级别，对大多数 action 来说，仅当 runlevel 和当前运行级别匹配成功才会执行。

3. action

action 是描述其后的 process 的运行方式的。action 可取的值包括：initdefault、sysinit、boot、bootwait 等：

initdefault 是个特别的 action 值，用于标识缺省的启动级别；当 init 由核心激活以后，他将读取 inittab 中的 initdefault 项，取得其中的 runlevel，并作为当前的运行级别。如果没有 inittab 文件，或其中没有 initdefault 项，init 将在控制台上请求输入 runlevel。

sysinit、boot、bootwait 等 action 将在系统启动时无条件运行，而忽略其中的 runlevel。

其余的 action（不含 initdefault）都和某个 runlevel 相关。各个 action 的定义在 inittab 的 man 手册中有周详的描述。

4. process

process 为具体的执行程序。程序后面能带参数。

第三部分：系统初始化

在 init 的设置文件中有这么一行：

```
si::sysinit:/etc/rc.d/rc.sysinit
```

他调用执行了/etc/rc.d/rc.sysinit，而 rc.sysinit 是个 bash shell 的脚本，他主要在 init 的设置文件中有这么一行：

```
si::sysinit:/etc/rc.d/rc.sysinit
```

他调用执行了/etc/rc.d/rc.sysinit，而 rc.sysinit 是个 bash shell 的脚本，他主要是完成一些系统初始化的工作，rc.sysinit 是每一个运行级别都要首先运行的重要脚本。他主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块及其他一些需要优先执行任务。

rc.sysinit 约有 850 多行，不过每个单一的功能还是比较简单，而且带有注释，建议有兴趣的用户能自行阅读自己机器上的该文件，以了解系统初始化所详情况。由于此文件较长，所以不在本文中列出来，也不做具体的介绍。

当 rc.sysinit 程序执行完毕后，将返回 init 继续下一步。

第四部分：启动对应运行级别的守护进程

在 rc.sysinit 执行后，将返回 init 继续其他的动作，通常接下来会执行到/etc/rc.d/rc 程序。以运行级别 5 为例，init 将执行设置文件 inittab 中的以下这行：

l5:5:wait:/etc/rc.d/rc 5

这一行表示以 5 为参数运行/etc/rc.d/rc，/etc/rc.d/rc 是个 Shell 脚本，他接受 5 作为参数，去执行/etc/rc.d/rc5.d/目录下的所有的 rc 启动脚本，/etc/rc.d/rc5.d/目录中的这些启动脚本实际上都是一些链接文件，而不是真正的 rc 启动脚本，真正的 rc 启动脚本实际上都是放在/etc/rc.d/init.d/目录下。而这些 rc 启动脚本有着类似的用法，他们一般能接受 start、stop、restart、status 等参数。

/etc/rc.d/rc5.d/中的 rc 启动脚本通常是 K 或 S 开头的链接文件，对于以 S 开头的启动脚本，将以 start 参数来运行。而如果发现存在相应的脚本也存在 K 打头的链接，而且已处于运行态了(以/var/lock/subsys/下的文件作为标志)，则将首先以 stop 为参数停止这些已启动了的守护进程，然后再重新运行。这样做是为了确保是当 init 改动运行级别时，所有相关的守护进程都将重启。

至于在每个运行级中将运行哪些守护进程，用户能通过 chkconfig 或 setup 中的"System Services"来自行设定。常见的守护进程有：

amd：自动安装 NFS 守护进程

apmd：高级电源管理守护进程

arpwatch：记录日志并构建一个在 LAN 接口上看到的以太网地址和 IP 地址对数据库

autofs：自动安装管理进程 automount，和 NFS 相关，依赖于 NIS

crond：Linux 下的计划任务的守护进程

named：DNS 服务器

netfs：安装 NFS、Samba 和 NetWare 网络文件系统

network：激活已设置网络接口的脚本程序

nfs：打开 NFS 服务

portmap：RPC portmap 管理器，他管理基于 RPC 服务的连接

sendmail：邮件服务器 sendmail

smb：Samba 文件共享/打印服务

syslog：一个让系统引导时起动 syslog 和 klogd 系统日志守候进程的脚本

xfs：X Window 字型服务器，为本地和远程 X 服务器提供字型集

Xinetd：支持多种网络服务的核心守护进程，能管理 wuftp、sshd、telnet 等服务

这些守护进程也启动完成了，rc 程序也就执行完了，然后又返回 init 继续下一步。

第五部分：建立终端

rc 执行完毕后，返回 init。这时基本系统环境已设置好了，各种守护进程也已启动了。init 接下来会打开 6 个终端，以便用户登录系统。通过按 Alt+Fn(n 对应 1-6)能在这 6 个终端中转换。在 inittab 中的以下 6 行就是定义了 6 个终端：

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

4:2345:respawn:/sbin/mingetty tty4

5:2345:respawn:/sbin/mingetty tty5

6:2345:respawn:/sbin/mingetty tty6

从上面能看出在 2、3、4、5 的运行级别中都将以 `respawn` 方式运行 `mingetty` 程序，`mingetty` 程序能打开终端、设置模式。同时他会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，而用户输入的用户将作为参数传给 `login` 程序来验证用户的身份。

第六部分：登录系统，启动完成

对于运行级别为 5 的图像方式用户来说，他们的登录是通过一个图像化的登录界面。登录成功后能直接进入 KDE、Gnome 等窗口管理器。而本文主要讲的还是文本方式登录的情况：

当我们看到 `mingetty` 的登录界面时，我们就能输入用户名和密码来登录系统了。

Linux 的账号验证程序是 `login`，`login` 会接收 `mingetty` 传来的用户名作为用户名参数。然后 `login` 会对用户名进行分析：如果用户名不是 `root`，且存在 `/etc/nologin` 文件，`login` 将输出 `nologin` 文件的内容，然后退出。这通常用来系统维护时防止非 `root` 用户登录。只有 `/etc/securetty` 中登记了的终端才允许 `root` 用户登录，如果不存在这个文件，则 `root` 能在所有终端上登录。`/etc /usertty` 文件用于对用户作出附加访问限制，如果不存在这个文件，则没有其他限制。

在分析完用户名后，`login` 将搜索 `/etc/passwd` 及 `/etc/shadow` 来验证密码及设置账户的其他信息，比如：主目录是什么、使用何种 `shell`。如果没有指定主目录，将默认为根目录；如果没有指定 `shell`，将默认为 `/bin/bash`。

`login` 程序成功后，会向对应的终端在输出最近一次登录的信息(在 `/var/log/lastlog` 中有记录)，并检查用户是否有新邮件(在 `/usr/spool/mail/` 的对应用户名目录下)。然后开始设置各种环境变量：对于 `bash` 来说，系统首先寻找 `/etc/profile` 脚本文件，并执行他；然后如果用户的主目录中存在 `.bash_profile` 文件，就执行他，在这些文件中又可能调用了其他设置文件，所有的设置文件执行后后，各种环境变量也设好了，这时会出现大家熟悉的命令行提示符，到此整个启动过程就结束了。

4、相关命令

4.1、查看当前系统运行等级

```
[root@test ~]# runlevel
N 5          // 'N'代表先前的 Runlevel; '5'代表目前的 Runlevel
```

4.2、切换系统运行等级

```
#init N      //切换到运行级别 N
# init 0     //关机
# init 6     //重新启动系统
```

十、进程管理

进程就是运行中的程序，一个运行着的程序，可能有多个进程。比如 `LinuxSir.Org` 所用的 `WWW` 服务器是 `apache` 服务器，当管理员启动服务后，可能会有好多人来访问，也就是说许多用户来同时请求 `httpd` 服务，`apache` 服务器将会创建有多个 `httpd` 进程来对其进行服务。

1、 进程分类

进程一般分为交互进程、批处理进程和守护进程三类。

值得一提的是守护进程总是活跃的，一般是后台运行，守护进程一般是由系统在开机时通过脚本自动激活启动或超级管理用户 `root` 来启动。比如在 `Fedora` 或 `Redhat` 中，我们可以定义 `httpd` 服务器的启动脚本的运行级别，此文件位于 `/etc/init.d` 目录下，文件名是 `httpd`，`/etc/init.d/httpd` 就是 `httpd` 服务器的守护程序，当把它的运行级别设置为 3 和 5 时，当系统启动时，它会跟着启动。

```
[root@localhost ~]# chkconfig --level 35 httpd on
```

由于守护进程是一直运行着的，所以它所处的状态是等待请求处理任务。比如，我们是不是访问 LinuxSir.Org，LinuxSir.Org 的 httpd 服务器都在运行，等待着用户来访问，也就是等待着任务处理。

2、进程的属性

进程 ID (PID): 是唯一的数值，用来区分进程；
父进程和父进程的 ID (PPID)；
启动进程的用户 ID (UID) 和所归属的组 (GID)；
进程状态：状态分为运行 R、休眠 S、僵尸 Z；
进程执行的优先级；
进程所连接的终端名；
进程资源占用：比如占用资源大小（内存、CPU 占用量）；

3、父进程和子进程

他们的关系是管理和被管理的关系，当父进程终止时，子进程也随之而终止。但子进程终止，父进程并不一定终止。比如 httpd 服务器运行时，我们可以杀掉其子进程，父进程并不会因为子进程的终止而终止。

在进程管理中，当我们发现占用资源过多，或无法控制的进程时，应该杀死它，以保护系统的稳定安全运行

4、进程管理命令

4.1、ps

ps 为我们提供了进程的一次性的查看，它所提供的查看结果并不动态连续的；如果想对进程时间监控，应该用 top 工具。

4.1.1、ps 的参数说明：

ps 提供了很多的选项参数，常用的有以下几个；

l 长格式输出；
u 按用户名和启动时间的顺序来显示进程；
j 用任务格式来显示进程；
f 用树形格式来显示进程；
a 显示所有用户的所有进程（包括其它用户）；
x 显示无控制终端的进程；
r 显示运行中的进程；
ww 避免详细参数被截断；
我们常用的选项是组合是 aux 或 lax，还有参数 f 的应用；
ps aux 或 lax 输出的解释；

```
[oracle@Test2 ~]$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	2072	632	?	Ss	Jun18	0:01	init [5]
root	2	0.0	0.0	0	0	?	S<	Jun18	0:00	[migration/0]

USER 表示启动进程用户。PID 表示进程标志号。%CPU 表示运行该进程占用 CPU 的时间与该进程总的运行时间的比例。%MEM 表示该进程占用内存和总内存的比例。VSZ 表示占用的虚拟内存大小，以 KB 为单位。RSS 为进程占用的物理内存值，以 KB 为单位。TTY 表示该进程建立时所对应的终端，"?"表示该进程不占用终端。STAT 表示进程的运行状态，包括以下几种代码：D，不可中断的睡眠；R，就绪（在可运行队列中）；S，睡眠；T，被跟踪或停止；Z，终止（僵死）的进程，Z 不

存在，但暂时无法消除；W，没有足够的内存分页可分配；<高优先序的进程；N，低优先序的进程；L，有内存分页分配并锁在内存体内（实时系统或 I/O）。START 为进程开始时间。TIME 为执行的时间。COMMAND 是对应的命令名。

4.1.2 ps 应用举例

实例一：ps aux 最常用

```
[root@localhost ~]# ps -aux | more
```

可以用 | 管道和 more 连接起来分页查看；

```
[root@localhost ~]# ps aux > ps001.txt
```

```
[root@localhost ~]# more ps001.txt
```

这里是把所有进程显示出来，并输出到 ps001.txt 文件，然后再通过 more 来分页查看；

实例二：和 grep 结合，提取指定程序的进程；

```
[root@localhost ~]# ps aux | grep httpd
```

root	4187	0.0	1.3	24236	10272 ?	Ss	11:55	0:00	/usr/sbin/httpd
apache	4189	0.0	0.6	24368	4940 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4190	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4191	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4192	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4193	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4194	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4195	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
apache	4196	0.0	0.6	24368	4932 ?	S	11:55	0:00	/usr/sbin/httpd
root	4480	0.0	0.0	5160	708 pts/3	R+	12:20	0:00	grep httpd

实例二：父进程和子进程的关系友好判断的例子

```
[root@localhost ~]# ps auxf | grep httpd
```

root	4484	0.0	0.0	5160	704 pts/3	S+	12:21	0:00	_ grep
httpd									
root	4187	0.0	1.3	24236	10272 ?	Ss	11:55	0:00	/usr/sbin/httpd
apache	4189	0.0	0.6	24368	4940 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4190	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4191	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4192	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4193	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4194	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4195	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd
apache	4196	0.0	0.6	24368	4932 ?	S	11:55	0:00	_ /usr/sbin/httpd

这里用到了 f 参数；父与子关系一目了然；

例三：找出消耗内存最多的前 10 名进程

```
# ps -auxf | sort -nr -k 4 | head -10
```

例四：找出使用 CPU 最多的前 10 名进程

```
# ps -auxf | sort -nr -k 3 | head -10
```

4.2、pstree

功能：pstree 命令列出当前的进程，以及它们的树状结构。

格式：pstree [选项] [pid|user]

主要选项如下：

- a: 显示执行程序的命令与完整参数。
- c: 取消同名程序，合并显示。
- h: 对输出结果进行处理，高亮显示正在执行的程序。
- l: 长格式显示。
- n: 以 PID 大小排序。
- p: 显示 PID。
- u: 显示 UID 信息。
- G: 使用 VT100 终端编码显示。
- U: 使用 UTF-8 (Unicode) 编码显示。

说明: 使用 `ps` 命令得到的数据精确, 但数据庞大, 这一点对掌握系统整体概况来说是不容易的。`pstree` 正好可以弥补这个缺憾。它可将当前的执行程序以树状结构显示。`pstree` 支持指定特定程序 (PID) 或使用者 (USER) 作为显示的起始。

应用实例如下。

进程启动的时候可能会产生自己的一个子进程。运行 `pstree` 命令就可以很容易地看到这些信息。以超级用户权限运行 `pstree`:

```
# init--apmd

|-atd

|-bdf flush

|-gconfd-2

|-gdm-binary---gdm-binary--X

|                                     `--startkde--kwrapper

|                                     `--ssh-agent

|-gpm

|-httpd---8*[httpd]

.....下略
```

命令对程序名称相同的会自动合并, 所有 "`| -httpd---8*[httpd]`" 即表示系统中有 8 个 `httpd` 进程产生的子进程。

4.3、top

`top` 命令用来显示系统当前的进程状况。

格式: `top [选项]`

主要选项如下。

- d: 指定更新的间隔, 以秒计算。
- q: 没有任何延迟的更新。如果使用者有超级用户, 则 `top` 命令将会以最高的优先序执行。
- c: 显示进程完整的路径与名称。
- S: 累积模式, 会将已完成或消失的子进程的 CPU 时间累积起来。

s: 安全模式。

i: 不显示任何闲置 (Idle) 或无用 (Zombie) 的进程。

n: 显示更新的次数, 完成后将会退出 top。

说明: top 命令和 ps 命令的基本作用是相同的, 都显示系统当前的进程状况。但是 top 是一个动态显示过程, 即可以通过用户按键来不断刷新当前状态。这里结合下图来说明它给出的信息。

```
top - 23:49:58 up 20:59, 3 users, load average: 0.03, 0.03, 0.00
Tasks: 114 total, 2 running, 110 sleeping, 0 stopped, 2 zombie
Cpu(s): 0.3%us, 1.7%sy, 0.0%ni, 98.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 299704k total, 295604k used, 4100k free, 68536k buffers
Swap: 524280k total, 13192k used, 511088k free, 98028k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
9614	root	14	-1	35956	10m	5944	S	1.3	3.4	0:30.11	X
9748	root	17	0	17252	2064	1724	S	0.3	0.7	0:01.48	escd
11209	root	15	0	2176	1024	800	R	0.3	0.3	0:00.40	top
1	root	15	0	2048	632	540	S	0.0	0.2	0:01.34	init
2	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
4	root	RT	0	0	0	0	S	0.0	0.0	0:00.04	watchdog/0
5	root	10	-5	0	0	0	S	0.0	0.0	0:00.94	events/0
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kthread
9	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	xenwatch
10	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	xenbus
12	root	10	-5	0	0	0	S	0.0	0.0	0:00.84	kblockd/0
13	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
71	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
75	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khubd
77	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kseriod

第一行表示的项目依次为当前时间、系统启动时间、当前系统登录用户数目、平均负载。

第二行显示的是 Tasks: 114 total 进程总数、2 running 正在运行的进程数、110 sleeping 睡眠的进程数、0 stopped 停止的进程数、2 zombie 僵尸进程数

第三行显示的是目前 CPU 的使用情况, Cpu(s): 0.3% us 用户空间占用 CPU 百分比、1.0% sy 内核空间占用 CPU 百分比、0.0% ni 用户进程空间内改变过优先级的进程占用 CPU 百分比、98.7% id 空闲 CPU 百分比、0.0% wa 等待输入输出的 CPU 时间百分比、0.0% hi、0.0% si

第四行显示物理内存的使用情况, Mem: 191272k total 物理内存总量、173656k used 使用的物理内存总量、17616k free 空闲内存总量、22052k buffers 用作内核缓存的内存量

第五行显示交换分区使用情况, Swap: 192772k total 交换区总量、0k used 使用的交换区总量、192772k free 空闲交换区总量、123988k cached 缓冲的交换区总量、内存中的内容被换出到交换区, 而后又被换入到内存, 但使用过的交换区尚未被覆盖, 该数值即为这些内容已存在于内存中的交换区的大小。相应的内存再次被换出时可不必再对交换区写入。

第六行显示的项目最多, 下面列出了详细解释。

PID (Process ID): 进程标志号, 是非零正整数。USER: 进程所有者的用户名。PR: 进程的优先级别。NI: 进程的优先级别数值。VIRT: 进程占用的虚拟内存值。RES: 进程占用的物理内存值。SHR: 进程使用的共享内存值。STAT: 进程的状态, 其中 S 表示休眠, R 表示正在运行, Z 表示僵死状态, N 表示该进程优先值是负数。%CPU: 该进程占用的 CPU 使用率。%MEM: 该进程占用的物理内存和总内

存的百分比。**TIME**: 该进程启动后占用的总的 CPU 时间。**COMMAND**: 进程启动的启动命令名称, 如果这一行显示不下, 进程会有一个完整的命令行。

top 命令使用过程中, 还可以使用一些交互的命令来完成其他参数的功能。这些命令是通过快捷键启动的。

<空格>: 立刻刷新。

A 分类显示系统不同资源的使用大户。有助于快速识别系统中资源消耗多的任务。

f 添加删除所要显示栏位。

o 调整所要显示栏位的顺序。

r 调整一个正在运行的进程 **Nice** 值。

k 结束一个正在运行的进程。

z 彩色/黑白显示开关

P: 根据 CPU 使用大小进行排序。

T: 根据时间、累计时间排序。

q: 退出 **top** 命令。

m: 切换显示内存信息。

t: 切换显示进程和 CPU 状态信息。

c: 切换显示命令名称和完整命令行。

M: 根据使用内存大小进行排序。

W: 将当前设置写入 `~/.toprc` 文件中。这是写 **top** 配置文件的推荐方法。

可以看到, **top** 命令是一个功能十分强大的监控系统的工具, 对于系统管理员而言尤其重要。但是, 它的缺点是会消耗很多系统资源。

十一、资源监控

1、free内存监控

语 法: `free [-bkmotV][-s <间隔秒数>]`

补充说明: **free** 指令会显示内存的使用情况, 包括实体内存, 虚拟的交换文件内存, 共享内存区段, 以及系统核心使用的缓冲区等。

参 数:

-b 以 Byte 为单位显示内存使用情况。

-k 以 KB 为单位显示内存使用情况。

-m 以 MB 为单位显示内存使用情况。

-o 不显示缓冲区调节列。

-s<间隔秒数> 持续观察内存使用状况。

-t 显示内存总和列。

-V 显示版本信息。

```
[root@test ~]# free -m
```

	total	used	free	shared	buffers	cached
Mem:	879	865	13	0	126	541
-/+ buffers/cache:		197	681			
Swap:	2047	0	2047			

Mem: 表示物理内存统计

-/+ buffers/cache: 表示物理内存的缓存统计

Swap: 表示硬盘上交换分区的使用情况

第 1 行 Mem: total: 表示物理内存总量。

used: 表示总计分配给缓存（包含 buffers 与 cache ）使用的数量，但其中可能部分缓存并未实际使用。

free: 未被分配的内存。

shared: 共享内存，一般系统不会用到，这里也不讨论。

buffers: 系统分配但未被使用的 buffers 数量。

cached: 系统分配但未被使用的 cache 数量。buffer 与 cache 的区别见后面。 $total = used + free$
第 2 行 $-/+ buffers/cached$: used: 也就是第一行中的 $used - buffers - cached$ 也是实际使用的内存总量。

free: 未被使用的 buffers 与 cache 和未被分配的内存之和，这就是系统当前实际可用内存。 $free2 = buffers1 + cached1 + free1 // free2$ 为第二行、 $buffers1$ 等为第一行

A buffer is something that has yet to be “written” to disk. A cache is something that has been “read” from the disk and stored for later use 第 3 行: 第三行所指的是从应用程序角度来看，对于应用程序来说， $buffers/cached$ 是等于可用的，因为 $buffer/cached$ 是为了提高文件读取的性能，当应用程序需在用到内存的时候， $buffer/cached$ 会很快地被回收。

所以从应用程序的角度来说，可用内存=系统 free memory+buffers+cached.

接下来解释什么时候内存会被交换，以及按什么方交换。

当可用内存少于额定值的时候，就会开会进行交换，如何看额定值（RHEL4.0）:

#cat /proc/meminfo

交换将通过三个途径来减少系统中使用的物理页面的个数:

- 1.减少缓冲与页面 cache 的大小，
- 2.将系统 V 类型的内存页面交换出去，
- 3.换出或者丢弃页面。(Application 占用的内存页，也就是物理内存不足)。

事实上，少量地使用 swap 是不是影响到系统性能的。

下面是 buffers 与 cached 的区别:

buffers 是指用来给块设备做的缓冲大小，他只记录文件系统的 metadata 以及 tracking in-flight pages.

cached 是用来给文件做缓冲。

那就是说: buffers 是用来存储，目录里面有什么内容，权限等等。

而 cached 直接用来记忆我们打开的文件，如果你想知道他是不是真的生效，你可以试一下，先后执行两次命令 `#man X`，你就可以明显的感觉到第二次的开打的速度快很多。

实验: 在一台没有什么应用的机器上做会看得比较明显。记得实验只能做一次，如果想多做请换一个文件名。

`#free`

`#man X`

`#free`

`#man X`

`#free`

你可以先后比较一下 free 后显示 buffers 的大小。

另一个实验:

`#free`

`#ls /dev`

`#free`

你比较一下两个的大小，当然这个 buffers 随时都在增加，但你有 ls 过的话，增加的速度会变得快，这个就是 buffers/chached 的区别。

因为 Linux 将你暂时不使用的内存作为文件和数据缓存, 以提高系统性能, 当你需要这些内存时, 系统会自动释放 (不像 windows 那样, 即使你有很多空闲内存, 他也要访问一下磁盘中的 pagefiles) 使用 free 命令

将 used 的值减去 buffer 和 cache 的值就是你当前真实内存使用 ———— - 对操作系统 来讲是 Mem 的参数. buffers/cached 都是属于被使用, 所以它认为 free 只有 16936 .

对应用程序 来讲是 (-/+ buffers/cach). buffers/cached 是等同可用的, 因为 buffer/cached 是为 了提高 程序执行的性能, 当程序使用内存时, buffer/cached 会很快地被使用。 所以, 以应用来看看, 以 (-/+ buffers/cache) 的 free 和 used 为主. 所以我们看这个就好了. 另外告诉大家 一些常识. Linux 为了提高磁盘和内存存取效率, Linux 做了很多精心的设计, 除了对 dentry 进行缓存 (用于 VFS, 加速文件路径名到 inode 的转换), 还采取了两种主要 Cache 方式: Buffer Cache 和 Page Cache。 前者针对磁盘块的读写, 后者针对文件 inode 的读写。这些 Cache 能有效缩短了 I/O 系统调用 (比如 read, write, getdents) 的时间。 记住内存是拿来用的, 不是拿来看的。 不象 windows, 无论你的真实物理内存有多少, 他都要拿硬盘交换 文件来读. 这也就是 windows 为什么常常提示虚拟空间不足的原因. 你们想想, 多无聊, 在内存还有大部分 的时候, 拿出一部分硬盘空间来充当内存. 硬盘怎么会快过内存. 所以我们看 linux, 只要不用 swap 的交换 空间, 就不用担心自己的内存太少. 如果常常 swap 用很多, 可能你就要考虑加物理内存了. 这也是 linux 看 内存是否够用的标准哦.

[root@scs-2 tmp]# free

	total	used	free	shared	buffers	cached
Mem:	3266180	3250004	16176		0	110652
-/+ buffers/cache:		471116	2795064			
Swap:	2048276	80160	1968116			

下面是对这些数值的解释:

total:总计物理内存的大小。

used:已使用多大。

free:可用有多少。

Shared:多个进程共享的内存总额。

Buffers/cached:磁盘缓存的大小。

第三行(-/+ buffers/cached):

used:已使用多大。

free:可用有多少。

第四行就不多解释了。

区别: 第二行(mem)的 used/free 与第三行(-/+ buffers/cache) used/free 的区别。 这两个的区别在于使用的角度来看, 第一行是从 OS 的角度来看, 因为对于 OS, buffers/cached 都是属于被使用, 所以他的可用内存是 16176KB, 已用内存是 3250004KB, 其中包括, 内核 (OS) 使用+Application(X, oracle, etc)使用的 +buffers+cached.

第三行所指的是从应用程序角度来看, 对于应用程序来说, buffers/cached 是等于可用的, 因为 buffer/cached 是为了提高文件读取的性能, 当应用程序需在用到内存的时候, buffer/cached 会很快地被回收。

所以从应用程序的角度来说, 可用内存=系统 free memory+buffers+cached。

如上例:

2795064=16176+110652+2668236

接下来解释什么时候内存会被交换, 以及按什么方交换。 当可用内存少于额定值的时候, 就会开会进行交换。

如何看额定值:

cat /proc/meminfo

[root@scs-2 tmp]# cat /proc/meminfo

MemTotal: 3266180 kB
MemFree: 17456 kB
Buffers: 111328 kB
Cached: 2664024 kB
SwapCached: 0 kB
Active: 467236 kB
Inactive: 2644928 kB
HighTotal: 0 kB
HighFree: 0 kB
LowTotal: 3266180 kB
LowFree: 17456 kB
SwapTotal: 2048276 kB
SwapFree: 1968116 kB
Dirty: 8 kB
Writeback: 0 kB
Mapped: 345360 kB
Slab: 112344 kB
Committed_AS: 535292 kB
PageTables: 2340 kB
VmallocTotal: 536870911 kB
VmallocUsed: 272696 kB
VmallocChunk: 536598175 kB
HugePages_Total: 0
HugePages_Free: 0
Hugepagesize: 2048 kB

用 free -m 查看的结果:

[root@scs-2 tmp]# free -m

	total	used	free	shared	buffers	cached
Mem:	3189	3173	16		0	107
-/+ buffers/cache:		460	2729			
Swap:	2000	78	1921			

查看/proc/kcore 文件的大小（内存镜像）:

[root@scs-2 tmp]# ll -h /proc/kcore

-r----- 1 root root 4.1G Jun 12 12:04 /proc/kcore

备注:

占用内存的测量

测量一个进程占用了多少内存，linux 为我们提供了一个很方便的方法，/proc 目录为我们提供了所有的信息，实际上 top 等工具也通过这里来获取相应的信息。

/proc/meminfo 机器的内存使用信息

/proc/pid/maps pid 为进程号，显示当前进程所占用的虚拟地址。

/proc/pid/statm 进程所占用的内存

```
[root@localhost ~]# cat /proc/self/statm
```

```
654 57 44 0 0 334 0
```

输出解释

CPU 以及 CPU0。。。的每行的每个参数意思（以第一行为例）为：

参数 解释 /proc//status

Size (pages) 任务虚拟地址空间的大小 VmSize/4

Resident(pages) 应用程序正在使用的物理内存的大小 VmRSS/4

Shared(pages) 共享页数 0

Trs(pages) 程序所拥有的可执行虚拟内存的大小 VmExe/4

Lrs(pages) 被映像到任务的虚拟内存空间的库的大小 VmLib/4

Drs(pages) 程序数据段和用户态的栈的大小 (VmData+ VmStk) 4

dt(pages) 04

查看机器可用内存

```
/proc/28248/>free
```

```
total used free shared buffers cached
```

```
Mem: 1023788 926400 97388 0 134668 503688
```

```
-/+ buffers/cache: 288044 735744
```

```
Swap: 1959920 89608 1870312
```

我们通过 free 命令查看机器空闲内存时，会发现 free 的值很小。这主要是因为，在 linux 中有这么一种思想，内存不用白不用，因此它尽可能的 cache 和 buffer 一些数据，以方便下次使用。但实际上这些内存也是可以立刻拿来使用的。

所以 空闲内存=free+buffers+cached=total-used

2、vmstat

很显然从名字中我们就可以知道 vmstat 是一个查看虚拟内存（Virtual Memory）使用状况的工具，但是怎样通过 vmstat 来发现系统中的瓶颈呢？在回答这个问题前，还是让我们回顾一下 Linux 中关于虚拟内存相关内容。

2.1、虚拟内存运行原理

在系统中运行的每个进程都需要使用到内存，但不是每个进程都需要每时每刻使用系统分配的内存空间。当系统运行所需内存超过实际的物理内存，内核会释放某些进程所占用但未使用的部分或所有物理内存，将这部分资料存储在磁盘上直到进程下一次调用，并将释放出的内存提供给有需要的进程使用。在 Linux 内存管理中，主要是通过“调页 Paging”和“交换 Swapping”来完成上述的内存调度。调页算法是将内存中最近不常使用的页面换到磁盘上，把活动页面保留在内存中供进程使用。交换技术是将整个进程，而不是部分页面，全部交换到磁盘上。

分页(Page)写入磁盘的过程被称作 Page-Out，分页(Page)从磁盘重新回到内存的过程被称作 Page-In。当内核需要一个分页时，却发现此分页不在物理内存中(因为已经被 Page-Out 了)，此时就发生了分页错误（Page Fault）。

当系统内核发现可运行内存变少时，就会通过 Page-Out 来释放一部分物理内存。经管 Page-Out 不是经常发生，但是如果 Page-out 频繁不断的发生，直到当内核管理分页的时间超过运行程式的时间时，系统效能会急剧下降。这时的系统已经运行非常慢或进入暂停状态，这种状态亦被称作 thrashing(颠簸)。

2.2、使用vmstat

1.用法

```
vmstat [-a] [-n] [-S unit] [delay [ count]]
```

```

vmstat [-s] [-n] [-S unit]
vmstat [-m] [-n] [delay [ count]]
vmstat [-d] [-n] [delay [ count]]
vmstat [-p disk partition] [-n] [delay [ count]]
vmstat [-f]
vmstat [-V]

```

-a: 显示活跃和非活跃内存

-f: 显示从系统启动至今的 fork 数量。

-m: 显示 slabinfo

-n: 只在开始时显示一次各字段名称。

-s: 显示内存相关统计信息及多种系统活动数量。

delay: 刷新时间间隔。如果不指定，只显示一条结果。

count: 刷新次数。如果不指定刷新次数，但指定了刷新时间间隔，这时刷新次数为无穷。

-d: 显示磁盘相关统计信息。

-p: 显示指定磁盘分区统计信息

-S: 使用指定单位显示。参数有 k、K、m、M，分别代表 1000、1024、1000000、1048576 字节 (byte)。默认单位为 K (1024 bytes)

-V: 显示 vmstat 版本信息。

2.3、实例

例子 1: 每 2 秒输出一条结果

```
[root@ ~]# vmstat 2
```

procs		memory				swap		io		system				cpu		
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	68	38636	162572	329964	0	0	0	3	1	0	0	0	100	0	0
0	0	68	38636	162572	329964	0	0	0	0	1022	337	0	1	99	0	0
0	0	68	38636	162572	329964	0	0	0	16	1024	332	0	0	100	0	0
0	0	68	38636	162572	329964	0	0	0	0	1020	338	0	1	99	0	0
0	0	68	38636	162572	329964	0	0	0	0	1024	330	0	0	100	0	0

<http://hi.baidu.com/imlidapeng>

字段说明:

Procs (进程):

r: 运行的和等待(CPU 时间片)运行的进程数，这个值也可以判断是否需要增加 CPU(长期大于 1)

b: 等待 IO 的进程数量，处于不可中断状态的进程数，常见的情况是由 IO 引起的

Memory (内存):

swpd: 使用虚拟内存大小，切换到交换内存上的内存(默认以 KB 为单位)

如果 swpd 的值不为 0，或者还比较大，比如超过 100M 了，但是 si,so 的值长期为 0，这种情况我们可以不用担心，不会影响系统性能。

free: 空闲的物理内存

buff: 用作缓冲的内存大小

cache: 用作缓存的内存大小，文件系统的 cache，如果 cache 的值大的时候，说明 cache 住的文件数多，如果频繁访问到的文件都能被 cache 住，那么磁盘的读 IO bi 会非常小

Swap:

si: 每秒从交换区写到内存的大小，交换内存使用，由磁盘调入内存

so: 每秒写入交换区的内存大小，交换内存使用，由内存调入磁盘

内存够用的时候，这 2 个值都是 0，如果这 2 个值长期大于 0 时，系统性能会受到影响。磁盘 IO 和 CPU 资源都会被消耗。

常有人看到空闲内存(free)很少或接近于 0 时, 就认为内存不够用了, 实际上不能光看这一点的, 还要结合 si,so, 如果 free 很少, 但是 si,so 也很少(大多数时候是 0), 那么不用担心, 系统性能这时不会受到影响的。

IO: (现在的 Linux 版本块的大小为 1024bytes)

bi: 每秒读取的块数, 从块设备读入的数据总量(读磁盘) (KB/s)

bo: 每秒写入的块数, 写入到块设备的数据总量(写磁盘) (KB/s)

随机磁盘读写的时候, 这 2 个 值越大 (如超出 1M), 能看到 CPU 在 IO 等待的值也会越大

system:

in: 每秒中断数, 包括时钟中断。

cs: 每秒上下文切换数。

上面这 2 个值越大, 会看到由内核消耗的 CPU 时间会越多

CPU (以百分比表示):

us: 用户进程消耗的 CPU 时间百分比, us 的值比较高时, 说明用户进程消耗的 CPU 时间多, 但是如果长期超过 50% 的使用, 那么我们就该考虑优化程序算法或者进行加速了

sy: 内核进程消耗的 CPU 时间百分比, sy 的值高时, 说明系统内核消耗的 CPU 资源多, 这并不是良性的表现, 我们应该检查原因。

id: CPU 处在空闲状态时间百分比(包括 IO 等待时间)

wa: IO 等待消耗的 CPU 时间百分比, wa 的值高时, 说明 IO 等待比较严重, 这可能是由于磁盘大量作随机访问造成, 也有可能是磁盘的带宽出现瓶颈(块操作)。

例子 2: 显示活跃和非活跃内存

```
[root@localhost ~]# vmstat -a 2
```

procs		memory				swap		io		system			cpu			
r	b	swpd	free	inact	active	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	68	38416	93472	831652	0	0	0	3	1	0	0	0	100	0	0
0	0	68	38416	93472	831652	0	0	0	0	1024	320	0	0	100	0	0
0	0	68	38416	93472	831652	0	0	0	0	1025	343	0	1	99	0	0
0	0	68	38112	93472	831692	0	0	0	50	1024	373	1	1	99	0	0
0	0	68	38112	93472	831692	0	0	0	0	1027	309	0	0	100	0	0

<http://hi.baidu.com/ymidapeng>

使用-a 选项显示活跃和非活跃内存时, 所显示的内容除增加 inact 和 active 外, 其他显示内容与例子 1 相同。

字段说明:

Memory (内存):

inact: 非活跃内存大小 (当使用-a 选项时显示)

active: 活跃的内存大小 (当使用-a 选项时显示)

注: 如果 r 经常大于 4, 且 id 经常少于 40, 表示 cpu 的负荷很重, 如果 bi, bo 长期不等于 0, 表示内存不足, 如果 disk 经常不等于 0, 且在 b 中的队列大于 3, 表示 io 性能不好。

CPU 问题现象:

- 1.)如果在 processes 中运行的序列(process r)是连续的大于在系统中的 CPU 的个数表示系统现在运行比较慢,有多数的进程等待 CPU.
- 2.)如果 r 的输出数大于系统中可用 CPU 个数的 4 倍的话,则系统面临着 CPU 短缺的问题,或者是 CPU 的速率过低,系统中有多数的进程在等待 CPU,造成系统中进程运行过慢.
- 3.)如果空闲时间(cpu id)持续为 0 并且系统时间(cpu sy)是用户时间的两倍(cpu us)系统则面临着 CPU 资源的短缺.

解决办法:

当发生以上问题的时候请先调整应用程序对 CPU 的占用情况.使得应用程序能够更有效的使用 CPU.同时可以考虑增加更多的 CPU. 关于 CPU 的使用情况还可以结合 mpstat, ps aux top prstat -a 等等一些相应的命令来综合考虑关于具体的 CPU 的使用情况,和那些进程在占用大量的 CPU 时间.一般情况下, 应用程序的问题会比较大一些.比如一些 SQL 语句不合理等等都会造成这样的现象.

内存问题现象:

内存的瓶颈是由 scan rate (sr)来决定的.scan rate 是通过每秒的始终算法来进行页扫描的.如果 scan rate(sr)连续的大于每秒 200 页则表示可能存在内存缺陷.同样的如果 page 项中的 pi 和 po 这两栏表示每秒页面的调入的页数和每秒调出的页数.如果该值经常为非零值,也有可能存在内存的瓶颈,当然,如果个别的时候不为 0 的话,属于正常的页面调度这个是虚拟内存的主要原理.

解决办法:

- 1.调节 applications & servers 使得对内存和 cache 的使用更加有效.
- 2.增加系统的内存.
3. Implement priority paging in s in pre solaris 8 versions by adding line "set priority paging=1" in /etc/system. Remove this line if upgrading from Solaris 7 to 8 & retaining old /etc/system file.

关于内存的使用情况还可以结 ps aux top prstat -a 等等一些相应的命令来综合考虑关于具体的内存的使用情况,和那些进程在占用大量的内存.一般情况下, 如果内存的占用率比较高,但是,CPU 的占用很低的时候,可以考虑是有很多的应用程序占用了内存没有释放,但是,并没有占用 CPU 时间,可以考虑应用程序,对于未占用 CPU 时间和一些后台的程序,释放内存的占用

4.4、案例

案例学习:

1: 在这个例子中,这个系统被充分利用

vmstat 1

procs				memory			swap			io		system				cpu	
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	wa	id		
3	0	206564	15092	80336	176080	0	0	0	0	718	26	81	19	0	0		
2	0	206564	14772	80336	176120	0	0	0	0	758	23	96	4	0	0		
1	0	206564	14208	80336	176136	0	0	0	0	820	20	96	4	0	0		
1	0	206956	13884	79180	175964	0	412	0	2680	1008	80	93	7	0	0		
2	0	207348	14448	78800	175576	0	412	0	412	763	70	84	16	0	0		
2	0	207348	15756	78800	175424	0	0	0	0	874	25	89	11	0	0		
1	0	207348	16368	78800	175596	0	0	0	0	940	24	86	14	0	0		
1	0	207348	16600	78800	175604	0	0	0	0	929	27	95	3	0	2		
3	0	207348	16976	78548	175876	0	0	0	2508	969	35	93	7	0	0		
4	0	207348	16216	78548	175704	0	0	0	0	874	36	93	6	0	1		
4	0	207348	16424	78548	175776	0	0	0	0	850	26	77	23	0	0		
2	0	207348	17496	78556	175840	0	0	0	0	736	23	83	17	0	0		
0	0	207348	17680	78556	175868	0	0	0	0	861	21	91	8	0	1		

根据观察值,我们可以得到以下结论:

- 1.有大量的中断(in) 和较少的上下文切换(cs).这意味着一个单一的进程在产生对硬件设备的请求.
- 2.进一步显示某单个应用,user time(us)经常在 85%或者更多.考虑到较少的上下文切换,这个应用应该还在处理器中被处理.
- 3.运行队列还在可接受的性能范围内,其中有 2 个地方,是超出了允许限制.

2: 在这个例子中,内核调度中的上下文切换处于饱和

vmstat 1

procs			memory				swap		io		system				cpu
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	wa	id
2	1	207740	98476	81344	180972	0	0	2496	0	900	2883	4	12	57	27
0	1	207740	96448	83304	180984	0	0	1968	328	810	2559	8	9	83	0
0	1	207740	94404	85348	180984	0	0	2044	0	829	2879	9	6	78	7
0	1	207740	92576	87176	180984	0	0	1828	0	689	2088	3	9	78	10
2	0	207740	91300	88452	180984	0	0	1276	0	565	2182	7	6	83	4
3	1	207740	90124	89628	180984	0	0	1176	0	551	2219	2	7	91	0
4	2	207740	89240	90512	180984	0	0	880	520	443	907	22	10	67	0
5	3	207740	88056	91680	180984	0	0	1168	0	628	1248	12	11	77	0
4	2	207740	86852	92880	180984	0	0	1200	0	654	1505	6	7	87	0
6	1	207740	85736	93996	180984	0	0	1116	0	526	1512	5	10	85	0
0	1	207740	84844	94888	180984	0	0	892	0	438	1556	6	4	90	0

根据观察值,我们可以得到以下结论:

- 1.上下文切换数目高于中断数目,说明 **kernel** 中相当数量的时间都开销在上下文切换线程.
- 2.大量的上下文切换将导致 **CPU** 利用率分类不均衡.很明显实际上等待 **io** 请求的百分比(**wa**)非常高,以及 **user time** 百分比非常低(**us**).
- 3.因为 **CPU** 都阻塞在 **IO** 请求上,所以运行队列里也有相当数目的可运行状态线程在等待执行.

3、iostat

用途: 报告中央处理器 (CPU) 统计信息和整个系统、适配器、tty 设备、磁盘和 CD-ROM 的输入 / 输出统计信息。

语法: `iostat [-c | -d] [-k] [-t | -m] [-V] [-x [device]] [interval [count]]`

描述: **iostat** 命令用来监视系统输入 / 输出设备负载, 这通过观察与它们的平均传送速率相关的物理磁盘的活动时间来实现。**iostat** 命令生成的报告可以用来更改系统配置来更好地平衡物理磁盘和适配器之间的输入 / 输出负载。

参数: **-c** 为汇报 CPU 的使用情况; **-d** 为汇报磁盘的使用情况; **-k** 表示每秒按 **kilobytes** 字节显示数据; **-m** 表示每秒按 **M** 字节显示数据; **-t** 为打印汇报的时间; **-v** 表示打印出版本信息和用法; **-x device** 指定要统计的设备名称, 默认为所有的设备; **interval** 指每次统计间隔的时间; **count** 指按照这个时间间隔统计的次数。

iostat 结果解析

```
[root@20081006-1724 ~]# iostat -x
Linux 2.6.9-78.ELsmp (20081006-1724) 11/20/2009
```

avg-cpu:	%user	%nice	%sys	%iowait	%idle										
	0.19	0.00	0.04	0.03	99.73										
Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	rkB/s	wkB/s	avgrq-sz	avgqu-sz	await	svctm	%util		
sda	0.05	17.60	1.46	7.72	80.69	202.57	40.34	101.29	30.87	0.01	1.06	0.37	0.34		
sda1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	29.90	0.00	3.14	3.14	0.00		
sda2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	16.25	0.00	1.51	1.30	0.00		
sda3	0.05	17.60	1.46	7.72	80.69	202.57	40.34	101.29	30.87	0.01	1.06	0.37	0.34		
dm-0	0.00	0.00	1.46	25.28	80.32	202.26	40.16	101.13	10.57	0.36	13.56	0.13	0.34		
dm-1	0.00	0.00	0.05	0.04	0.37	0.32	0.18	0.16	8.00	0.00	6.84	1.30	0.01		

rrqm/s: 每秒进行 **merge** 的读操作数目。即 $\text{delta}(\text{rmerge})/\text{s}$

wrqm/s: 每秒进行 **merge** 的写操作数目。即 $\text{delta}(\text{wmerge})/\text{s}$

r/s: 每秒完成的读 I/O 设备次数。即 $\text{delta}(\text{rio})/\text{s}$

w/s: 每秒完成的写 I/O 设备次数。即 $\text{delta}(\text{wio})/\text{s}$

rsec/s: 每秒读扇区数。即 $\text{delta}(\text{rsect})/\text{s}$

wsect/s: 每秒写扇区数。即 $\text{delta(wsect)}/s$

rkB/s: 每秒读 K 字节数。是 rsect/s 的一半, 因为每扇区大小为 512 字节。

wkB/s: 每秒写 K 字节数。是 wsect/s 的一半。

avgrq-sz: 平均每次设备 I/O 操作的数据大小 (扇区)。即 $\text{delta(rsect+wsect)}/\text{delta(rio+wio)}$

avgqu-sz: 平均 I/O 队列长度。即 $\text{delta(aveq)}/s/1000$ (因为 aveq 的单位为毫秒)。

await: 平均每次设备 I/O 操作的等待时间 (毫秒)。即 $\text{delta(ruse+wuse)}/\text{delta(rio+wio)}$

svctm: 平均每次设备 I/O 操作的服务时间 (毫秒)。即 $\text{delta(use)}/\text{delta(rio+wio)}$

%util: 一秒中有百分之多少的时间用于 I/O 操作, 或者说一秒中有多少时间 I/O 队列是非空的。即 $\text{delta(use)}/s/1000$ (因为 use 的单位为毫秒)

如果 %util 接近 100%, 说明产生的 I/O 请求太多, I/O 系统已经满负荷, 该磁盘可能存在瓶颈。

比较重要的参数

%util: 一秒中有百分之多少的时间用于 I/O 操作, 或者说一秒中有多少时间 I/O 队列是非空的

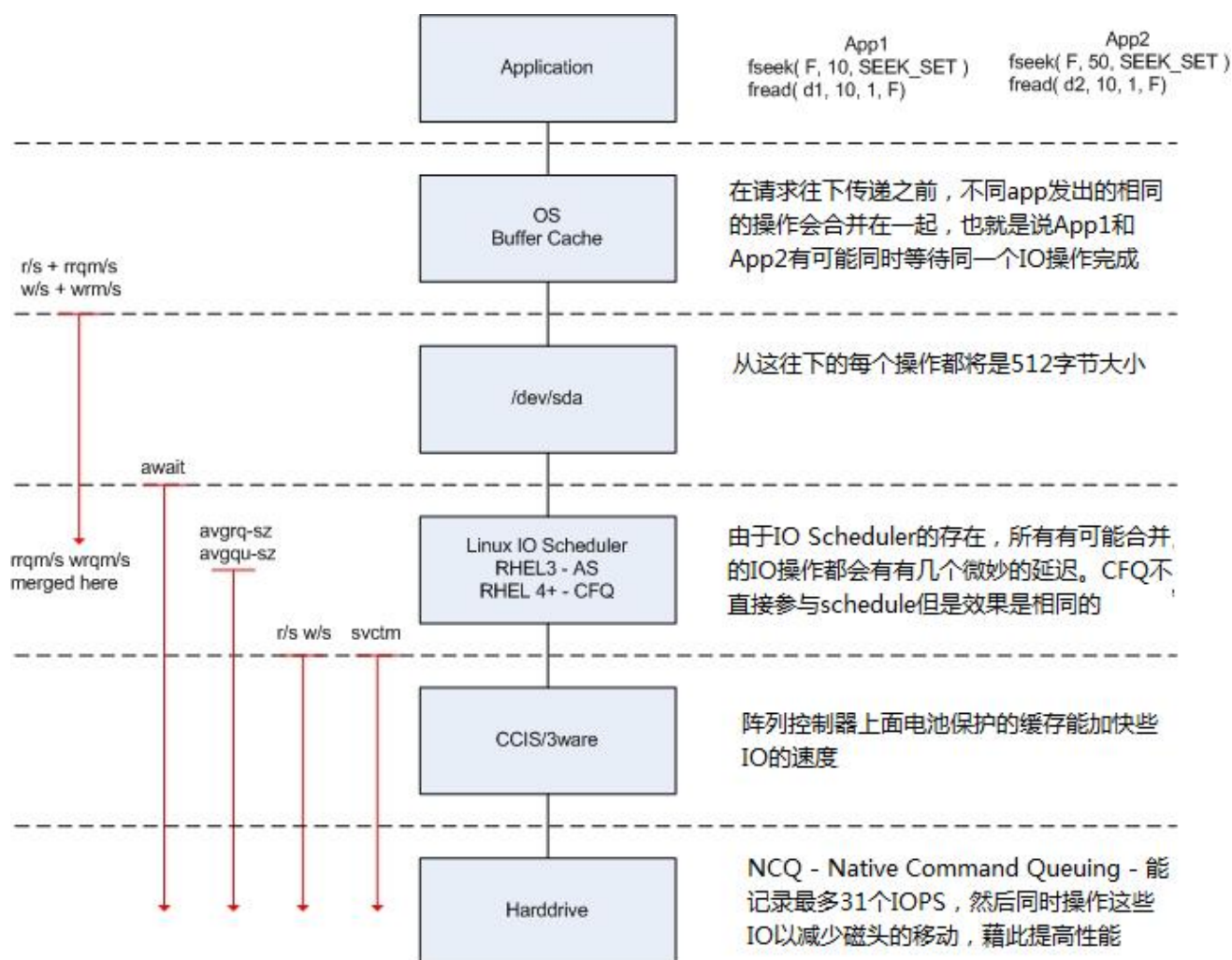
svctm: 平均每次设备 I/O 操作的服务时间

await: 平均每次设备 I/O 操作的等待时间

avgqu-sz: 平均 I/O 队列长度

如果 %util 接近 100%, 表明 i/o 请求太多, i/o 系统已经满负荷, 磁盘可能存在瓶颈, 一般 %util 大于 70%, i/o 压力就比较大, 读取速度有较多的 wait。同时可以结合 vmstat 查看查看 b 参数(等待资源的进程数)和 wa 参数 (IO 等待所占用的 CPU 时间的百分比, 高过 30% 时 IO 压力高)。

要理解这些性能指标我们先看下图



IO 的执行过程的各个参数

上图的左边是 `iostat` 显示的各个性能指标，每个性能指标都会显示在一条虚线之上，这表明这个性能指标是从虚线之上的那个读写阶段开始计量的，比如说图中的 `w/s` 从 Linux IO scheduler 开始穿过硬盘控制器(CCIS/3ware)，这就表明 `w/s` 统计的是每秒钟从 Linux IO scheduler 通过硬盘控制器的写 IO 的数量。

结合上图对读 IO 操作的过程做一个说明，在从 OS Buffer Cache 传入到 OS Kernel(Linux IO scheduler) 的读 IO 操作的个数实际上是 `rrqm/s+r/s`，直到读 IO 请求到达 OS Kernel 层之后，有每秒钟有 `rrqm/s` 个读 IO 操作被合并，最终转送给磁盘控制器的每秒钟读 IO 的个数为 `r/w`；在进入到操作系统的设备层(`/dev/sda`)之后，计数器开始对 IO 操作进行计时，最终的计算结果表现是 `await`，这个值就是我们想要的 IO 响应时间了；`svctm` 是在 IO 操作进入到磁盘控制器之后直到磁盘控制器返回结果所花费的时间，这是一个实际 IO 操作所花的时间，当 `await` 与 `svctm` 相差很大的时候，我们就要注意磁盘的 IO 性能了；而 `avgrq-sz` 是从 OS Kernel 往下传递请求时单个 IO 的大小，`avgqu-sz` 则是在 OS Kernel 中 IO 请求队列的平均大小。

现在我们可以将 `iostat` 输出结果和我们上面讨论的指标挂钩了

设备 IO 操作:总 IO(io)/s = r/s(读) + w/s(写) = 1.46 + 25.28 = 26.74

平均每次设备 I/O 操作只需要 0.36 毫秒完成,现在却需要 10.57 毫秒完成,因为发出的请求太多(每秒 26.74 个),假如请求同时发出的,可以这样计算平均等待时间:

平均等待时间=单个 I/O 服务器时间*(1+2+...+请求总数-1)/请求总数

每秒发出的 I/O 请求很多,但是平均队列就 4,表示这些请求比较均匀,大部分处理还是比较及时

`svctm` 一般要小于 `await` (因为同时等待的请求的等待时间被重复计算了), `svctm` 的大小一般和磁盘性能有关,CPU/内存的负荷也会对其有影响,请求过多也会间接导致 `svctm` 的增加。`await` 的大小一般取决于服务时间(`svctm`) 以及 I/O 队列的长度和 I/O 请求的发出模式。如果 `svctm` 比较接近 `await`, 说明 I/O 几乎没有等待时间;如果 `await` 远大于 `svctm`,说明 I/O 队列太长,应用得到的响应时间变慢,如果响应时间超过了用户可以容许的范围,这时可以考虑更换更快的磁盘,调整内核 `elevator` 算法,优化应用,或者升级 CPU。

队列长度(`avgqu-sz`)也可作为衡量系统 I/O 负荷的指标,但由于 `avgqu-sz` 是按照单位时间的平均值,所以不能反映瞬间的 I/O 洪水。

I/O 系统 vs. 超市排队

举一个例子,我们在超市排队 checkout 时,怎么决定该去哪个交款台呢?首当是看排的队人数,5 个人总比 20 人要快吧?除了数人头,我们也常常看看前面人购买的东西多少,如果前面有个采购了一星期食品的大妈,那么可以考虑换个队排了。还有就是收银员的速度了,如果碰上了连钱都点不清楚的新手,那就有的

等了。另外,时机也很重要,可能 5 分钟前还人满为患的收款台,现在已是人去楼空,这时候交款可是很爽啊,当然,前提是那过去的 5 分钟里所做的事情比排队要有意义(不过我还没发现什么事情比排队还无聊的)。

I/O 系统也和超市排队有很多类似之处:

`r/s+w/s` 类似于交款人的总数

平均队列长度(`avgqu-sz`)类似于单位时间里平均排队人的个数

平均服务时间(`svctm`)类似于收银员的收款速度

平均等待时间(`await`)类似于平均每人的等待时间

平均 I/O 数据(`avgrq-sz`)类似于平均每人所买的东西多少

I/O 操作率 (%util)类似于收款台前有人排队的时间比例。

我们可以根据这些数据分析出 I/O 请求的模式,以及 I/O 的速度和响应时间。

一个例子

```
# iostat -x 1
avg-cpu:  %user  %nice  %sys  %idle
16.24 0.00 4.31 79.44
Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s kB/s kB/s avgrq-sz avgqu-sz await svctm  %util
/dev/cciss/c0d0
0.00 44.90 1.02 27.55 8.16 579.59 4.08 289.80 20.57 22.35 78.21 5.00 14.29
/dev/cciss/c0d0p1
0.00 44.90 1.02 27.55 8.16 579.59 4.08 289.80 20.57 22.35 78.21 5.00 14.29
/dev/cciss/c0d0p2
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

上面的 iostat 输出表明秒有 28.57 次设备 I/O 操作: $\text{delta}(\text{io})/\text{s} = \text{r/s} + \text{w/s} = 1.02 + 27.55 = 28.57$ (次/秒) 其中写操作占了主体 (w:r = 27:1)。

平均每次设备 I/O 操作只需要 5ms 就可以完成, 但每个 I/O 请求却需要等上 78ms, 为什么? 因为发出的 I/O 请求太多 (每秒钟约 29 个), 假设这些请求是同时发出的, 那么平均等待时间可以这样计算: 平均等待时间 = 单个 I/O 服务时间 * (1 + 2 + ... + 请求总数 - 1) / 请求总数

应用到上面的例子: 平均等待时间 = $5\text{ms} * (1 + 2 + \dots + 28) / 29 = 70\text{ms}$, 和 iostat 给出的 78ms 的平均等待时间很接近。这反过来表明 I/O 是同时发起的。每秒发出的 I/O 请求很多 (约 29 个), 平均队列却不长 (只有 2 个左右), 这表明这 29 个请求的到来并不均匀, 大部分时间 I/O 是空闲的。一秒中有 14.29% 的时间 I/O 队列中是有请求的, 也就是说, 85.71% 的时间里 I/O 系统无事可做, 所有 29 个 I/O 请求都在 142 毫秒之内处理掉了。

$\text{delta}(\text{ruse} + \text{wuse}) / \text{delta}(\text{io}) = \text{await} = 78.21 \Rightarrow \text{delta}(\text{ruse} + \text{wuse})/\text{s} = 78.21 * \text{delta}(\text{io})/\text{s} = 78.21 * 28.57 = 2232.8$, 表明每秒内的 I/O 请求总共需要等待 2232.8ms。所以平均队列长度应为 $2232.8\text{ms} / 1000\text{ms} = 2.23$, 而 iostat

给出的平均队列长度 (avgqu-sz) 却为 22.35, 为什么?! 因为 iostat 中有 bug, avgqu-sz 值应为 2.23, 而不是 22.35。

4、uptime

uptime 命令用于查看服务器运行了多长时间以及有多少个用户登录, 快速获知服务器的负荷情况。uptime 的输出包含一项内容是 load average, 显示了最近 1, 5, 15 分钟的负荷情况。它的值代表等待 CPU 处理的进程数, 如果 CPU 没有时间处理这些进程, load average 值会升高; 反之则会降低。load average 的最佳值是 1, 说明每个进程都可以马上处理并且没有 CPU cycles 被丢失。对于单 CPU 的机器, 1 或者 2 是可以接受的值; 对于多路 CPU 的机器, load average 值可能在 8 到 10 之间。也可以使用 uptime 命令来判断网络性能。例如, 某个网络应用性能很低, 通过运行 uptime 查看服务器的负荷是否很高, 如果不是, 那么问题应该是网络方面造成的。

以下是 uptime 的运行实例:

```
9:24am up 19:06, 1 user, load average: 0.00, 0.00, 0.00
```

也可以查看 /proc/loadavg 和 /proc/uptime 两个文件, 注意不能编辑 /proc 中的文件, 要用 cat 等命令来查看, 如:

```
liyawei:~ # cat /proc/loadavg
```

```
0.0 0.00 0.00 1/55 5505
```

uptime 命令用法十分简单: 直接输入

```
# uptime
```

例:

```
18:02:41 up 41 days, 23:42, 1 user, load average: 0.00, 0.00, 0.00
```


1 可以被认为是最优的负载值。负载是会随着系统不同改变得。单 CPU 系统 1-3 和 SMP 系统 6-10 都是可能接受的。

另外还有一个参数 `-v`，是用来查询版本的。（注意是大写的字母 v）

```
[linux @ localhost]$ uptime -v
```

```
procps version 3.2.7
```

```
[linux @ localhost]$ uptime
```

显示结果为：

```
10:19:04 up 257 days, 18:56, 12 users, load average: 2.10, 2.10, 2.09
```

显示内容说明：

```
10:19:04 //系统当前时间
```

```
up 257 days, 18:56 //主机已运行时间,时间越大，说明你的机器越稳定。
```

```
12 user //用户连接数，是总连接数而不是用户数
```

```
load average // 系统平均负载，统计最近 1， 5， 15 分钟的系统平均负载
```

那么什么是系统平均负载呢？系统平均负载是指在特定时间间隔内运行队列中的平均进程数。如果每个 CPU 内核的当前活动进程数不大于 3 的话，那么系统的性能是良好的。如果每个 CPU 内核的任务数大于 5，那么这台机器的性能有严重问题。如果你的 linux 主机是 1 个双核 CPU 的话，当 Load Average 为 6 的时候说明机器已经被充分使用了。

5、W

w 命令主要是查看当前登录的用户，这个命令相对来说比较简单。我们来看一下截图。

```
[root@dherrup-us-pc ~]# w
03:08:21 up 13 min, 1 user, load average: 1.03, 1.13, 0.79
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU WHAT
root      pts/2    :0            03:00       0.00s       0.05s       0.00s w
```

在上面这个截图里面呢，第一列 user，代表登录的用户，第二列，tty 代表用户登录的终端号，因为在 linux 中并不是只有一个终端的，pts/2 代表是图形界面的第二个终端（这仅是个人意见，网上的对 pts 的看法可能有些争议）。第三列 FROM 代表登录的地方，如果是远程登录的，会显示 ip 地址，:0 表示的是 display 0:0，意思就是主控制台的第一个虚拟终端。第四列 login@ 代表登录的时间，第五列的 IDLE 代表系统的闲置时间。最后一列 what 是代表正在运行的进程，因为我正在运行 w 命令，所以咋 root 显示 w。

5、mpstat

mpstat （RHEL5 默认不安装）

mpstat 是 MultiProcessor Statistics 的缩写，是实时系统监控工具。其报告与 CPU 的一些统计信息，这些信息存放在 /proc/stat 文件中。在多 CPUs 系统里，其不但能查看所有 CPU 的平均状况信息，而且能够查看特定 CPU 的信息。下面只介绍 mpstat 与 CPU 相关的参数，mpstat 的语法如下：

```
mpstat [-P {ALL}] [internal [count]]
```

参数的含义如下：

-P {ALL} 表示监控哪个 CPU，cpu 在 [0,cpu 个数-1] 中取值

internal 相邻的两次采样的间隔时间

count 采样的次数，count 只能和 delay 一起使用

当没有参数时，mpstat 则显示系统启动以后所有信息的平均值。有 interval 时，第一行的信息自系统启动以来的平均信息。

从第二行开始，输出为前一个 interval 时间段的平均信息。与 CPU 有关的输出的含义如下：

```
[oracle@Test ~]$ mpstat -P ALL
```

Linux 2.6.18-194.el5 (Test.linux.com) 2010 年 06 月 22 日

09 时 18 分 18 秒	CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
09 时 18 分 18 秒	all	0.06	0.00	0.43	0.78	0.00	0.00	0.00	98.71	1069.35
09 时 18 分 18 秒	0	0.05	0.00	0.36	0.17	0.02	0.00	0.00	99.41	1032.01
09 时 18 分 18 秒	1	0.04	0.00	0.42	0.07	0.00	0.00	0.00	99.47	0.26
09 时 18 分 18 秒	2	0.11	0.00	0.28	0.08	0.00	0.00	0.00	99.52	0.00
09 时 18 分 18 秒	3	0.07	0.00	0.48	0.05	0.00	0.00	0.00	99.39	0.01
09 时 18 分 18 秒	4	0.08	0.00	0.19	5.63	0.00	0.02	0.00	94.08	24.51
09 时 18 分 18 秒	5	0.05	0.00	0.63	0.11	0.00	0.00	0.00	99.21	0.22
09 时 18 分 18 秒	6	0.07	0.00	0.45	0.10	0.00	0.01	0.00	99.36	12.33
09 时 18 分 18 秒	7	0.05	0.00	0.64	0.07	0.00	0.00	0.00	99.24	0.00

参数 解释 从/proc/stat 获得数据 CPU 处理器 ID

user 在 internal 时间段里, 用户的 CPU 时间 (%), 不包含 nice 值为负 进程 (usr/total)*100

nice 在 internal 时间段里, nice 值为负进程的 CPU 时间 (%) (nice/total)*100

system 在 internal 时间段里, 核心时间 (%) (system/total)*100

iowait 在 internal 时间段里, 硬盘 IO 等待时间 (%) (iowait/total)*100

irq 在 internal 时间段里, 硬中断时间 (%) (irq/total)*100

soft 在 internal 时间段里, 软中断时间 (%) (softirq/total)*100

idle 在 internal 时间段里, CPU 除去等待磁盘 IO 操作外的因为任何原因而空闲的时间闲置时间 (%) (idle/total)*100

intr/s 在 internal 时间段里, 每秒 CPU 接收的中断的次数 intr/total)*100

CPU 总的工作时间=total_cur=user+system+nice+idle+iowait+irq+softirq

total_pre=pre_user+ pre_system+ pre_nice+ pre_idle+ pre_iowait+ pre_irq+ pre_softirq

user=user_cur - user_pre

total=total_cur-total_pre

其中_cur 表示当前值, _pre 表示 interval 时间前的值。上表中的所有值可取到两位小数点。

范例 1: average mode (粗略信息)当 mpstat 不带参数时, 输出为从系统启动以来的平均值。

```
[work@builder linux-2.6.14]$ mpstat
```

Linux 2.6.9-5.31AXsmp (builder.redflag-linux.com) 12/16/2005

09:38:46 AM	CPU	%user	%nice	%system	%iowait	%irq	%soft	%idle	intr/s
-------------	-----	-------	-------	---------	---------	------	-------	-------	--------

09:38:48 AM	all	23.28	0.00	1.75	0.50	0.00	0.00	74.47	1018.59
-------------	-----	-------	------	------	------	------	------	-------	---------

范例 2: 每 2 秒产生了 2 个处理器的统计数据报告

下面的命令可以每 2 秒产生了 2 个处理器的统计数据报告, 一共产生三个 interval 的信息, 然后再给出这三个 interval 的平均信息。默认时, 输出是按照 CPU 号排序。第一个行给出了从系统引导以来的所有活跃数据。接下来每行对应一个处理器的活跃状态。

```
[root@server yum_dir]# mpstat -P ALL 2 3
```

Linux 2.6.18-164.el5 (server.sys.com) 01/04/2010

09:34:20 PM	CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
09:34:22 PM	all	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	1001.49
09:34:22 PM	0	0.00	0.00	0.50	0.00	0.00	0.00	0.00	99.50	1001.00
09:34:22 PM	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	0.00
09:34:22 PM	CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
09:34:24 PM	all	0.00	0.00	0.25	0.00	0.00	0.00	0.00	99.75	1005.00
09:34:24 PM	0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00	1005.50


```

09:34:24 PM      1      0.00      0.00      0.00      0.00      0.00      0.00      0.00 100.00      0.00
09:34:24 PM CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
09:34:26 PM all    0.00    0.00    0.00    0.00    0.00    0.00    0.00 100.00    1001.49
09:34:26 PM      0      0.00      0.00      0.00      0.00      0.00      0.00      0.00 100.00    1001.00
09:34:26 PM      1      0.00      0.00      0.00      0.00      0.00      0.00      0.00 100.00      0.00
Average:        CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
Average:        all    0.00    0.00    0.08    0.00    0.00    0.00    0.00 99.92    1002.66
Average:         0      0.00    0.00    0.17    0.00    0.00    0.00    0.00 99.83    1002.49
Average:         1      0.00    0.00    0.00    0.00    0.00    0.00    0.00 100.00      0.00

```

范例 3：比较带参数和不带参数的 mpstat 的结果。

在后台开一个 2G 的文件

```
# cat 1.img &
```

然后在另一个终端运行 mpstat 命令

```
[root@server ~]# cat 1.img &
```

```
[1] 6934
```

```
[root@server ~]# mpstat
```

```
Linux 2.6.18-164.el5 (server.sys.com)      01/04/2010
```

```

10:17:31 PM CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
10:17:31 PM all    0.07    0.02    0.25    0.21    0.01    0.04    0.00 99.40    1004.57

```

```
[root@server ~]# mpstat
```

```
Linux 2.6.18-164.el5 (server.sys.com)      01/04/2010
```

```

10:17:35 PM CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
10:17:35 PM all    0.07    0.02    0.25    0.21    0.01    0.04    0.00 99.39    1004.73

```

```
[root@server ~]# mpstat
```

```
Linux 2.6.18-164.el5 (server.sys.com)      01/04/2010
```

```

10:17:39 PM CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
10:17:39 PM all    0.07    0.02    0.25    0.21    0.01    0.04    0.00 99.38    1004.96

```

```
[root@server ~]# mpstat
```

```
Linux 2.6.18-164.el5 (server.sys.com)      01/04/2010
```

```

10:17:44 PM CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
10:17:44 PM all    0.07    0.02    0.26    0.21    0.01    0.05    0.00 99.37    1005.20

```

```
[root@server ~]# mpstat 3 10
```

```
Linux 2.6.18-164.el5 (server.sys.com)      01/04/2010
```

```

10:17:55 PM CPU    %user    %nice    %sys %iowait    %irq    %soft %steal    %idle    intr/s
10:17:58 PM all    13.12    0.00    20.93    0.00    1.83    9.80    0.00 54.32    2488.08
10:18:01 PM all    10.82    0.00    19.30    0.83    1.83    9.32    0.00 57.90    2449.83
10:18:04 PM all    10.95    0.00    20.40    0.17    1.99    8.62    0.00 57.88    2384.05
10:18:07 PM all    10.47    0.00    18.11    0.00    1.50    8.47    0.00 61.46    2416.00
10:18:10 PM all    11.81    0.00    22.63    0.00    1.83    11.98    0.00 51.75    2210.60
10:18:13 PM all     6.31    0.00    10.80    0.00    1.00    5.32    0.00 76.58    1795.33
10:18:19 PM all     1.75    0.00     3.16    0.75    0.25    1.25    0.00 92.85    1245.18
10:18:22 PM all    11.94    0.00    19.07    0.00    1.99    8.29    0.00 58.71    2630.46
10:18:25 PM all    11.65    0.00    19.30    0.50    2.00    9.15    0.00 57.40    2673.91
10:18:28 PM all    11.44    0.00    21.06    0.33    1.99    10.61    0.00 54.56    2369.87

```

Average: all 9.27 0.00 16.18 0.30 1.50 7.64 0.00 65.11 2173.54

[root@server ~]#

上两表显示出当要正确反映系统的情况，需要正确使用命令的参数。vmstat 和 iostat 也需要注意这一问题。

6、pmap

pmap 命令可以显示进程的内存映射，使用这个命令可以找出造成内存瓶颈的原因。

pmap -d PID

显示 PID 为 47394 进程的内存信息。

pmap -d 47394

输出样例：

47394: /usr/bin/php-cgi

Address	Kbytes	Mode	Offset	Device	Mapping
0000000000400000	2584	r-x--	0000000000000000	008:00002	php-cgi
0000000000886000	140	rw---	0000000000286000	008:00002	php-cgi
00000000008a9000	52	rw---	00000000008a9000	000:00000	[anon]
0000000000aa8000	76	rw---	00000000002a8000	008:00002	php-cgi
0000000000f678000	1980	rw---	0000000000f678000	000:00000	[anon]
000000314a600000	112	r-x--	0000000000000000	008:00002	ld-2.5.so
000000314a81b000	4	r----	000000000001b000	008:00002	ld-2.5.so
000000314a81c000	4	rw---	000000000001c000	008:00002	ld-2.5.so
000000314aa00000	1328	r-x--	0000000000000000	008:00002	libc-2.5.so
000000314ab4c000	2048	-----	000000000014c000	008:00002	libc-2.5.so
.....					
00002af8d48fd000	4	rw---	0000000000006000	008:00002	xsl.so
00002af8d490c000	40	r-x--	0000000000000000	008:00002	libnss_files-2.5.so
00002af8d4916000	2044	-----	000000000000a000	008:00002	libnss_files-2.5.so
00002af8d4b15000	4	r----	0000000000009000	008:00002	libnss_files-2.5.so
00002af8d4b16000	4	rw---	000000000000a000	008:00002	libnss_files-2.5.so
00002af8d4b17000	768000	rw-s-	0000000000000000	000:00009	zero (deleted)
00007fffc95fe000	84	rw---	00007fffffea000	000:00000	[stack]
ffffffff600000	8192	-----	0000000000000000	000:00000	[anon]
mapped: 933712K writeable/private: 4304K shared: 768000K					

最后一行非常重要：

* mapped: 933712K 内存映射所占空间大小

* writeable/private: 4304K 私有地址空间大小

* shared: 768000K 共享地址空间大小

7、sar

sar 是一个优秀的一般性能监视工具，它可以输出 Linux 所完成的几乎所有工作的数据。sar 命令在 sysstat rpm 中提供。示例中使用 sysstat 版本 5.0.5，这是稳定的最新版本之一。关于版本和下载信息，请访问 sysstat 主页 <http://perso.wanadoo.fr/sebastien.godard/>。

sar 可以显示 CPU、运行队列、磁盘 I/O、分页（交换区）、内存、CPU 中断、网络等性能数据。最重要的 sar 功能是创建数据文件。每一个 Linux 系统都应该通过 cron 工作收集 sar 数据。该 sar 数据文件为

系统管理员提供历史性能信息。这个功能非常重要，它将 **sar** 和其他性能工具区分开。如果一个夜晚批处理工作正常运行两次，直到下一个早上才会发现这种情况（除非被叫醒）。我们需要具备研究 12 小时以前的性能数据的能力。**sar** 数据收集器提供了这种能力。

```
sar 命令行的常用格式：
sar [options] [-A] [-o file] t [n]
```

在命令行中，**n** 和 **t** 两个参数组合起来定义采样间隔和次数，**t** 为采样间隔，是必须有的参数，**n** 为采样次数，是可选的，默认值是 1，**-o file** 表示将命令结果以二进制格式存放在文件中，**file** 在此处不是关键字，是文件名。**options** 为命令行选项，**sar** 命令的选项很多，下面只列出常用选项：

- A: 所有报告的总和。
- u: CPU 利用率
- v: 进程、I 节点、文件和锁表状态。
- d: 硬盘使用报告。
- r: 没有使用的内存页面和硬盘块。
- g: 串口 I/O 的情况。
- b: 缓冲区使用情况。
- a: 文件读写情况。
- c: 系统调用情况。
- R: 进程的活动情况。
- y: 终端设备活动情况。
- w: 系统交换活动。

7.1、CPU统计数据

sar -u 输出显示 CPU 信息。**-u** 选项是 **sar** 的默认选项。该输出以百分比显示 CPU 的使用情况。表 3-2 解释该输出。

表 3-2 sar -u 字段

字 段	说 明
CPU	CPU 编号
%user	CPU 花费在用户进程（如应用程序、Shell 脚本或与该用户进行的交互）上的时间的百分比。
%nice	运行正常进程所花的时间
%system	在内核模式（系统）中运行进程所花的时间，也就是 CPU 用来执行核心任务的时间的百分比。
%iowait	没有进程在该 CPU 上执行时，处理器等待 I/O 完成的时间
%idle	没有进程在该 CPU 上执行的时间

这些看起来应该比较熟悉，它和 **top** 报告中的 CPU 信息内容相同。以下显示输出格式：

```
[oracle@Test ~]$ sar -u -o aixi 60 5
Linux 2.6.18-194.el5 (Test.linux.com) 06/22/10
13:41:25      CPU      %user      %nice      %system      %iowait      %steal      %idle
13:42:25    all        0.28        0.00        0.21        1.17        0.00      98.34
13:43:25    all        0.23        0.00        0.16        1.14        0.00      98.46
```

13:44:25	all	0.27	0.00	0.21	1.40	0.00	98.12
13:45:25	all	0.26	0.00	0.19	0.99	0.00	98.56
13:46:25	all	0.32	0.00	0.23	1.39	0.00	98.05
Average:	all	0.27	0.00	0.20	1.22	0.00	98.31

每 60 秒采样一次，连续采样 5 次，观察 CPU 的使用情况，并将采样结果以二进制形式存入当前目录下的文件 aixi 中

在所有的显示中，我们应主要注意%wio 和%idle，%wio 的值过高，表示硬盘存在 I/O 瓶颈，%idle 值高，表示 CPU 较空闲，如果%idle 值高但系统响应慢时，有可能是 CPU 等待分配内存，此时应加大内存容量。%idle 值如果持续低于 10，那么系统的 CPU 处理能力相对较低，表明系统中最需要解决的资源是 CPU。另外任何 sar 报告的第一列都是时间戳。

如果要查看二进制文件 aixi 中的内容，则需键入如下 sar 命令：

```
# sar -u -f aixi
```

可见，sar 命令即可以实时采样，又可以对以往的采样结果进行查询。

我们本来可以研究使用-f 选项通过 sadc 创建的文件。这个 sar 语法显示 sar -f/var/log/ sa/sa21 的输出：

```
[dave@fisher dave]$ sar -f /var/log/sa/sa21|head -n 20
Linux 2.4.21-20.EL (fisher) 04/21/2005
```

12:00:00 AM	CPU	%user	%nice	%system	%iowait	%idle
12:10:00 AM	all	0.23	0.00	0.23	0.02	99.52
12:20:00 AM	all	0.22	0.00	0.20	0.01	99.57
12:30:01 AM	all	0.21	0.00	0.19	0.01	99.59
12:40:00 AM	all	0.23	0.00	0.22	0.02	99.54
12:50:01 AM	all	0.19	0.00	0.28	0.01	99.52
01:00:00 AM	all	0.22	0.00	0.18	0.01	99.59
01:10:00 AM	all	0.40	0.00	0.25	0.02	99.34
01:20:00 AM	all	0.20	0.00	0.25	0.01	99.53
01:30:00 AM	all	0.20	0.00	0.23	0.01	99.56
01:40:00 AM	all	0.22	0.00	0.21	0.02	99.56
01:50:00 AM	all	0.22	0.00	0.20	0.02	99.56
02:00:00 AM	all	0.22	0.00	0.19	0.01	99.58
02:10:00 AM	all	0.23	0.00	0.24	0.02	99.50
02:20:01 AM	all	0.22	0.00	0.25	0.01	99.52
02:30:00 AM	all	0.19	0.00	0.22	0.01	99.57
02:40:00 AM	all	0.22	0.00	0.22	0.01	99.55
02:50:00 AM	all	0.21	0.00	0.21	0.01	99.56

在多 CPU Linux 系统中，sar 命令也可以为每个 CPU 分解该信息，如以下 sar -u -P ALL 5 5 输出所示：

```
Linux 2.4.21-20.ELsmp (doughboy)      04/13/2005
```

10:17:56 AM	CPU	%user	%nice	%system	%iowait	%idle
10:18:01 AM	all	26.41	0.00	23.46	0.00	50.13
10:18:01 AM	0	23.20	0.00	32.80	0.00	44.00
10:18:01 AM	1	26.60	0.00	20.80	0.00	52.60
10:18:01 AM	2	25.80	0.00	21.00	0.00	53.20
10:18:01 AM	3	30.06	0.00	19.24	0.00	50.70

10:18:01 AM	CPU	%user	%nice	%system	%iowait	%idle
10:18:06 AM	all	17.70	0.00	24.15	0.00	58.15
10:18:06 AM	0	22.40	0.00	26.40	0.00	51.20
10:18:06 AM	1	15.20	0.00	24.60	0.00	60.20
10:18:06 AM	2	19.00	0.00	20.00	0.00	61.00
10:18:06 AM	3	14.20	0.00	25.60	0.00	60.20

10:18:06 AM	CPU	%user	%nice	%system	%iowait	%idle
10:18:11 AM	all	13.69	0.00	23.74	0.05	62.52
10:18:11 AM	0	9.00	0.00	27.40	0.00	63.60
10:18:11 AM	1	19.40	0.00	20.40	0.20	60.00
10:18:11 AM	2	13.20	0.00	21.00	0.00	65.80
10:18:11 AM	3	13.17	0.00	26.15	0.00	60.68

10:18:11 AM	CPU	%user	%nice	%system	%iowait	%idle
10:18:16 AM	all	16.40	0.00	23.00	0.00	60.60
10:18:16 AM	0	16.60	0.00	18.00	0.00	65.40
10:18:16 AM	1	15.00	0.00	23.00	0.00	62.00
10:18:16 AM	2	19.40	0.00	19.80	0.00	60.80
10:18:16 AM	3	14.60	0.00	31.20	0.00	54.20

10:18:16 AM	CPU	%user	%nice	%system	%iowait	%idle
10:18:21 AM	all	32.60	0.00	22.10	0.00	45.30
10:18:21 AM	0	30.80	0.00	24.40	0.00	44.80
10:18:21 AM	1	34.80	0.00	24.00	0.00	41.20
10:18:21 AM	2	32.00	0.00	20.20	0.00	47.80
10:18:21 AM	3	32.80	0.00	19.80	0.00	47.40

Average:	CPU	%user	%nice	%system	%iowait	%idle
Average:	all	21.36	0.00	23.29	0.01	55.34
Average:	0	20.40	0.00	25.80	0.00	53.80
Average:	1	22.20	0.00	22.56	0.04	55.20
Average:	2	21.88	0.00	20.40	0.00	57.72
Average:	3	20.96	0.00	24.40	0.00	54.64

磁盘I/O统计数据

sar 是一个研究磁盘 I/O 的优秀工具。以下是 sar 磁盘 I/O 输出的一个示例。

```
[dave@fisher dave]$ sar -d 5 2
Linux 2.4.21-27.EL (fisher)      04/29/2005
```

04:22:03 PM	DEV	tps	rd_sec/s	wr_sec/s
04:22:08 PM	dev3-0	1.40	0.00	784.00
04:22:08 PM	dev3-1	0.00	0.00	0.00
04:22:08 PM	dev3-2	1.40	0.00	784.00
04:22:08 PM	dev3-3	0.00	0.00	0.00
04:22:08 PM	dev3-64	0.00	0.00	0.00
04:22:08 PM	dev3-65	0.00	0.00	0.00
04:22:08 PM	DEV	tps	rd_sec/s	wr_sec/s
04:22:13 PM	dev3-0	34.60	0.00	4219.20
04:22:13 PM	dev3-1	0.00	0.00	0.00
04:22:13 PM	dev3-2	34.60	0.00	4219.20
04:22:13 PM	dev3-3	0.00	0.00	0.00
04:22:13 PM	dev3-64	0.00	0.00	0.00
04:22:13 PM	dev3-65	0.00	0.00	0.00
Average:	DEV	tps	rd_sec/s	wr_sec/s
Average:	dev3-0	18.00	0.00	2501.60
Average:	dev3-1	0.00	0.00	0.00
Average:	dev3-2	18.00	0.00	2501.60
Average:	dev3-3	0.00	0.00	0.00
Average:	dev3-64	0.00	0.00	0.00
Average:	dev3-65	0.00	0.00	0.00

第一行-d 显示磁盘 I/O 信息，5 2 选项是间隔和迭代，就像 sar 数据收集器那样。表 3-3 列出了字段和说明。

表 3-3 sar -d 字段

字 段	说 明
DEV	磁盘设备
tps	每秒传输数（或者每秒 IO 数）
rd_sec/s	每秒 512 字节读取数
wr_sec/s	每秒 512 字节写入数

512 只是一个测量单位，不表示所有磁盘 I/O 均使用 512 字节块。DEV 列是 dev#-#格式的磁盘设备，其中第一个#是设备主编号，第二个#是次编号或者连续编号。对于大于 2.5 的内核，sar 使用次编号。例如，在 sar -d 输出中看到的 dev3-0 和 dev3-1。它们对应于/dev/hda 和/dev/hda1。请看/dev 中的以下各项：

```
brw-rw---- 1 root disk 3, 0 Jun 24 2004 hda
brw-rw---- 1 root disk 3, 1 Jun 24 2004 hda1
```

/dev/hda 有主编号 3 和次编号 0。hda1 有主编号 3 和次编号 1。

3.2.4 网络统计数据

sar 提供四种不同的语法选项来显示网络信息。-n 选项使用四个不同的开关：DEV、EDEV、SOCK 和 FULL。DEV 显示网络接口信息，EDEV 显示关于网络错误的统计数据，SOCK 显示套接字信息，FULL 显示所有三个开关。它们可以单独或者一起使用。表 3-4 显示通过-n DEV 选项报告的字段。

表 3-4 sar -n DEV 字段

字 段	说 明
IFACE	LAN 接口


```
# sar -n EDEV 5 3
Linux 2.4.21-20.EL (fisher)    04/17/2005

10:41:44 AM      IFACE  rxerr/s  txerr/s    coll/s  rxdrop/s  txdrop/s
txcarr/s rxfram/s rxfifo/s txfifo/s
10:41:49 AM      lo      0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00
10:41:49 AM      eth0    0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00

10:41:49 AM      IFACE  rxerr/s  txerr/s    coll/s  rxdrop/s  txdrop/s
txcarr/s rxfram/s rxfifo/s txfifo/s
10:41:54 AM      lo      0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00
10:41:54 AM      eth0    0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00

10:41:54 AM      IFACE  rxerr/s  txerr/s    coll/s  rxdrop/s  txdrop/s
txcarr/s rxfram/s rxfifo/s txfifo/s
10:41:59 AM      lo      0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00
10:41:59 AM      eth0    0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00

Average:          IFACE  rxerr/s  txerr/s    coll/s  rxdrop/s  txdrop/s
txcarr/s rxfram/s rxfifo/s txfifo/s
Average:          lo      0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00
Average:          eth0    0.00    0.00      0.00    0.00    0.00
0.00    0.00    0.00    0.00
```

表 3-5 sar -n EDEV 字段

字 段	说 明
IFACE	LAN 接口
rxerr/s	每秒钟接收的坏数据包
txerr/s	每秒钟发送的坏数据包
coll/s	每秒冲突数
rxdrop/s	因为缓冲充满，每秒钟丢弃的已接收数据包数
txdrop/s	因为缓冲充满，每秒钟丢弃的已发送数据包数
txcarr/s	发送数据包时，每秒载波错误数
rxfram/s	每秒接收数据包的帧对齐错误数
rxfifo/s	接收的数据包每秒 FIFO 过速的错误数
txfifo/s	发送的数据包每秒 FIFO 过速的错误数

SOCK 参数显示 IPCS 套接字信息。表 3-6 列出显示的字段及其意义。

表 3-6 sar -n SOCK 字段

字 段	说 明
totsck	使用的套接字总数量
tcpsock	使用的 TCP 套接字数量
udpsock	使用的 UDP 套接字数量
rawsock	使用的 raw 套接字数量
ip-frag	使用的 IP 段数量


```
# sar -n SOCK 5 3
Linux 2.4.21-144-default (sawnee)      04/17/05

16:00:56      totsck      tcpsck      udpsck      rawsck      ip-frag
16:01:01          117          11           8           0           0
16:01:06          117          11           8           0           0
16:01:11          117          11           8           0           0
Average:          117          11           8           0           0
```

sar 可以产生许多其他报告。我们有必要仔细阅读 sar (1) 手册页，查看是否有自己需要的其他报告

例二：使用命令行 `sar -v t n`

例如，每 30 秒采样一次，连续采样 5 次，观察核心表的状态，需键入如下命令：

```
# sar -v 30 5
```

屏幕显示：

```
SC0_SV scosysv 3.2v5.0.5 i80386 10/01/2001
10:33:23 proc-sz ov inod-sz ov file-sz ov lock-sz      (-v)
10:33:53 305/ 321  0 1337/2764  0 1561/1706  0 40/ 128
10:34:23 308/ 321  0 1340/2764  0 1587/1706  0 37/ 128
10:34:53 305/ 321  0 1332/2764  0 1565/1706  0 36/ 128
10:35:23 308/ 321  0 1338/2764  0 1592/1706  0 37/ 128
10:35:53 308/ 321  0 1335/2764  0 1591/1706  0 37/ 128
```

显示内容包括：

proc-sz：目前核心中正在使用或分配的进程表的表项数，由核心参数 MAX-PROC 控制。

inod-sz：目前核心中正在使用或分配的 i 节点表的表项数，由核心参数 MAX-INODE 控制。

file-sz：目前核心中正在使用或分配的文件表的表项数，由核心参数 MAX-FILE 控制。

ov：溢出出现的次数。

Lock-sz：目前核心中正在使用或分配的记录加锁的表项数，由核心参数 MAX-FLCKRE 控制。

显示格式为

实际使用表项/可以使用的表项数

显示内容表示，核心使用完全正常，三个表没有出现溢出现象，核心参数不需调整，如果出现溢出时，要调整相应的核心参数，将对应的表项数加大。

例三：使用命令 `sar -d t n`

例如，每 30 秒采样一次，连续采样 5 次，报告设备使用情况，需键入如下命令：

```
# sar -d 30 5
```

屏幕显示：

```
SCO_SV scosysv 3.2v5.0.5 i80386 10/01/2001
11:06:43 device %busy      avque      r+w/s      blks/s      await avserv (-d)
11:07:13 wd-0    1.47       2.75       4.67       14.73       5.50 3.14
11:07:43 wd-0    0.43       18.77      3.07       8.66       25.11 1.41
11:08:13 wd-0    0.77       2.78       2.77       7.26       4.94 2.77
11:08:43 wd-0    1.10       11.18      4.10       11.26      27.32 2.68
11:09:13 wd-0    1.97       21.78      5.86       34.06      69.66 3.35
Average wd-0    1.15       12.11      4.09       15.19      31.12 2.80
```

显示内容包括：

device: sar 命令正在监视的块设备的名字。
%busy: 设备忙时，传送请求所占时间的百分比。
avque: 队列站满时，未完成请求数量的平均值。
r+w/s: 每秒传送到设备或从设备传出的数据量。
blks/s: 每秒传送的块数，每块 512 字节。
await: 队列占满时传送请求等待队列空闲的平均时间。
avserv: 完成传送请求所需平均时间（毫秒）。

在显示的内容中，wd-0 是硬盘的名字，%busy 的值比较小，说明用于处理传送请求的有效时间太少，文件系统效率不高，一般来讲，%busy 值高些，avque 值低些，文件系统的效率比较高，如果%busy 和 avque 值相对比较高，说明硬盘传输速度太慢，需调整。

例四：使用命令 `sar -b t n`

例如，每 30 秒采样一次，连续采样 5 次，报告缓冲区的使用情况，需键入如下命令：

```
# sar -b 30 5
```

屏幕显示：

```
SCO_SV scosysv 3.2v5.0.5 i80386 10/01/2001
14:54:59 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s (-b)
14:55:29 0      147      100      5      21      78      0      0
14:55:59 0      186      100      5      25      79      0      0
```

14:56:29	4	232	98	8	58	86	0	0
14:56:59	0	125	100	5	23	76	0	0
14:57:29	0	89	100	4	12	66	0	0
Average	1	156	99	5	28	80	0	0

显示内容包括：

bread/s: 每秒从硬盘读入系统缓冲区 buffer 的物理块数。
 lread/s: 平均每秒从系统 buffer 读出的逻辑块数。
 %rcache: 在 buffer cache 中进行逻辑读的百分比。
 bwrit/s: 平均每秒从系统 buffer 向磁盘所写的物理块数。
 lwrit/s: 平均每秒写到系统 buffer 逻辑块数。
 %wcache: 在 buffer cache 中进行逻辑读的百分比。
 pread/s: 平均每秒请求物理读的次数。
 pwrit/s: 平均每秒请求物理写的次数。

在显示的内容中,最重要的是%cache 和%wcache 两列,它们的值体现着 buffer 的使用效率,%rcache 的值小于 90 或者%wcache 的值低于 65,应适当增加系统 buffer 的数量,buffer 数量由核心参数 NBUF 控制,使%rcache 达到 90 左右,%wcache 达到 80 左右。但 buffer 参数值的多少影响 I/O 效率,增加 buffer,应在较大内存的情况下,否则系统效率反而得不到提高。

例五：使用命令 `sar -g t n`

例如,每 30 秒采样一次,连续采样 5 次,报告串口 I/O 的操作情况,需键入如下命令:

```
# sar -g 30 5
```

屏幕显示:

```
SC0_SV scosysv 3.2v5.0.5 i80386 11/22/2001
17:07:03 ovsiohw/s ovsiodma/s ovclist/s (-g)
17:07:33 0.00 0.00 0.00
17:08:03 0.00 0.00 0.00
17:08:33 0.00 0.00 0.00
17:09:03 0.00 0.00 0.00
17:09:33 0.00 0.00 0.00
Average 0.00 0.00 0.00
```

显示内容包括:

ovsiohw/s: 每秒在串口 I/O 硬件出现的溢出。

ovsiodma/s: 每秒在串口 I/O 的直接输入输出通道高速缓存出现的溢出。

ovclist/s : 每秒字符队列出现的溢出。

在显示的内容中，每一列的值都是零，表明在采样时间内，系统中没有发生串口 I/O 溢出现象。

sar 命令的用法很多，有时判断一个问题，需要几个 sar 命令结合起来使用，比如，怀疑 CPU 存在瓶颈，可用 sar -u 和 sar -q 来看，怀疑 I/O 存在瓶颈，可用 sar -b、sar -u 和 sar -d 来看。

Sar

- A 所有的报告总和
- a 文件读，写报告
- B 报告附加的 buffer cache 使用情况
- b buffer cache 使用情况
- c 系统调用使用报告
- d 硬盘使用报告
- g 有关串口 I/O 情况
- h 关于 buffer 使用统计数字
- m IPC 消息和信号灯活动
- n 命名 cache
- p 调页活动
- q 运行队列和交换队列的平均长度
- R 报告进程的活动
- r 没有使用的内存页面和硬盘块
- u CPU 利用率
- v 进程，i 节点，文件和锁表状态
- w 系统交换活动
- y TTY 设备活动

-a 报告文件读，写报告

sar -a 5 5

SC0_SV scosvr 3.2v5.0.5 PentII (D) ISA 06/07/2002

11:45:40 iget/s namei/s dirbk/s (-a)

11:45:45 6 2 2

11:45:50 91 20 28

11:45:55 159 20 18

11:46:00 157 21 19

11:46:05 177 30 35

Average 118 18 20

iget/s 每秒由 i 节点项定位的文件数量

namei/s 每秒文件系统路径查询的数量

dirbk/s 每秒所读目录块的数量

* 这些值越大,表明核心花在存取用户文件上的时间越多,它反映着一些程序和应用文件系统产生的负荷。一般地,如果 `iget/s` 与 `namei/s` 的比值大于 5, 并且 `namei/s` 的值大于 30, 则说明文件系统是低效的。这时需要检查文件系统的自由空间,看看是否自由空间过少。

-b 报告缓冲区 (buffer cache) 的使用情况

```
sar -b 2 3
SC0_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/07/2002
13:51:28 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s (-b)
13:51:30 382 1380 72 131 273 52 0 0
13:51:32 378 516 27 6 22 72 0 0
13:51:34 172 323 47 39 57 32 0 0
Average 310 739 58 58 117 50 0 0
```

`bread/s` 平均每秒从硬盘 (或其它块设备) 读入系统 buffer 的物理块数

`lread/s` 平均每秒从系统 buffer 读出的逻辑块数

`%rcache` 在 buffer cache 中进行逻辑读的百分比 (即 $100\% - \text{bread}/\text{lreads}$)

`bwrit/s` 平均每秒从系统 buffer 向磁盘 (或其它块设备) 所写的物理块数

`lwrit/s` 平均每秒写到系统 buffer 的逻辑块数

`%wcache` 在 buffer cache 中进行逻辑写的百分比 (即 $100\% - \text{bwrit}/\text{lwrit}$) .

`pread/sgu` 平均每秒请求进行物理读的次数

`pwrit/s` 平均每秒请求进行物理写的次数

* 所显示的内容反映了目前与系统 buffer 有关的读,写活。在所报告的数字中,最重要的是 `%rcache` 和 `%wcache` (统称为 cache 命中率) 两列,它们具体体现着系统 buffer 的效率。衡量 cache 效率的标准是它的命中率值的大小。

* 如果 `%rcache` 的值小于 90 或者 `%wcache` 的值低于 65, 可能就需要增加系统 buffer 的数量。如果在系统的应用中,系统的 I/O 活动十分频繁,并且在内存容量配置比较大时,可以增加 buffer cache, 使 `%rcache` 达到 95 左右, `%wcache` 达到 80 左右。

* 系统 buffer cache 中,buffer 的数量由核心参数 `NBUF` 控制。它是一个要调的参数。系统中 buffer 数量的多少是影响系统 I/O 效率的瓶颈。要增加系统 buffer 数量,则要求应该有较大的内存配置。否则一味增加 buffer 数量,势必减少用户进程在内存中的运行空间,这同样会导致系统效率下降。

-c 报告系统调用使用情况

```
sar -c 2 3
SC0_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/07/2002
17:02:42 scall/s sread/s swrit/s fork/s exec/s rchar/s wchar/s (-c)
17:02:44 2262 169 141 0.00 0.00 131250 22159
17:02:46 1416 61 38 0.00 0.00 437279 6464
17:02:48 1825 43 25 0.00 0.00 109397 42331
Average 1834 91 68 0.00 0.00 225975 23651
```

`scall/s` 每秒使用系统调用的总数。一般地,当 4~6 个用户在系统上工作时,每秒大约 30 个左右。

sread/s 每秒进行读操作的系统调用数量。

swrit/s 每秒进行写操作的系统调用数量。

fork/s 每秒 fork 系统调用次数。当 4~6 个用户在系统上工作时，每秒大约 0.5 秒左右。

exec/s 每秒 exec 系统调用次数。

rchar/s 每秒由读操作的系统调用传送的字符（以字节为单位）。

wchar/s 每秒由写操作的系统调用传送的字符（以字节为单位）。

* 如果 scall/s 持续地大于 300，则表明正在系统中运行的可能是效率很低的应用程序。在比较典型的情况下，进行读操作的系统调用加上进行写操作的系统调用之和，约是 scall 的一半左右。

-d 报告硬盘使用情况

```
sar -d 2 3
```

```
SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/07/2002
```

```
17:27:49 device %busy avque r+w/s blks/s await avserv (-d)
```

```
17:27:51 ida-0 6.93 1.00 13.86 259.41 0.00 5.00
```

```
ida-1 0.99 1.00 17.33 290.10 0.00 0.57
```

```
17:27:53 ida-0 75.50 1.00 54.00 157.00 0.00 13.98
```

```
ida-1 9.50 1.00 12.00 75.00 0.00 7.92
```

```
17:27:55 ida-0 7.46 1.00 46.77 213.93 0.00 1.60
```

```
ida-1 17.41 1.00 57.71 494.53 0.00 3.02
```

```
Average ida-0 29.85 1.00 38.14 210.28 0.00 7.83
```

```
ida-1 9.29 1.00 29.02 286.90 0.00 3.20
```

device 这是 sar 命令正在监视的块设备的名字。

%busy 设备忙时，运行传送请求所占用的时间。这个值以百分比表示。

avque 在指定的时间周期内，没有完成的请求数量的平均值。仅在队列被占满时取这个值。

r+w/s 每秒传送到设备或者从设备传送出的数据量。

blks/s 每秒传送的块数。每块 512 个字节。

await 传送请求等待队列空闲的平均时间（以毫秒为单位）。仅在队列被占满时取这个值。

avserv 完成传送请求所需平均时间（以毫秒为单位）

* ida-0 和 ida-1 是硬盘的设备名字。在显示的内容中，如果 %busy 的值比较小，说明用于处理传送请求的有效时间太少，文件系统的效率不高。要使文件系统的效率得到优化，应使 %busy 的数值相对高一些，而 avque 的值应该低一些。

-g 报告有关串口 I/O 情况

```
sar -g 3 3
```

```
SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/13/2002
```

```
11:10:09 ovsiohw/s ovsiodma/s ovclist/s (-g)
```

```
11:10:12 0.00 0.00 0.00
```

```
11:10:15 0.00 0.00 0.00
```

```
11:10:18 0.00 0.00 0.00
```

```
Average 0.00 0.00 0.00
```

ovsiow/s 每秒在串口 I/O 硬件出现的溢出。

ovsiDMA/s 每秒在串口 I/O 的直接输入，输出信道高速缓存出现的溢出。

ovclist/s 每秒字符队列出现的溢出。

-m 报告进程间的通信活动（IPC 消息和信号灯活动）情况

```
sar -m 4 3
```

SC0_SV scosvr 3.2v5.0.5 PentII (D) ISA 06/13/2002

13:24:28 msg/s sema/s (-m)

13:24:32 2.24 9.95

13:24:36 2.24 21.70

13:24:40 2.00 36.66

Average 2.16 22.76

msg/s 每秒消息操作的次数（包括发送消息的接收信息）。

sema/s 每秒信号灯操作次数。

* 信号灯和消息作为进程间通信的工具，如果在系统中运行的应用过程中没有使用它们，那么由 sar 命令报告的 msg 和 sema 的值都将等于 0.00。如果使用了这些工具，并且其中或者 msg/s 大于 100，或者 sema/s 大于 100，则表明这样的应用程序效率比较低。原因是在这样的应用程序中，大量的时间花费在进程之间的沟通上，而对保证进程本身有效的运行时间必然产生不良的影响。

-n 报告命名缓冲区活动情况

```
sar -n 4 3
```

SC0_SV scosvr 3.2v5.0.5 PentII (D) ISA 06/13/2002

13:37:31 c_hits cmisses (hit %) (-n)

13:37:35 1246 71 (94%)

13:37:39 1853 81 (95%)

13:37:43 969 56 (94%)

Average 1356 69 (95%)

c_hits cache 命中的数量。

cmisses cache 未命中的数量。

(hit %) 命中数量/(命中数量+未命中数量)。

* 不难理解，(hit %) 值越大越好，如果它低于 90%，则应该调整相应的核心参数。

-p 报告分页活动

```
sar -p 5 3
```

SC0_SV scosvr 3.2v5.0.5 PentII (D) ISA 06/13/2002

13:45:26 vflt/s pflt/s pgfil/s rclm/s (-p)

13:45:31 36.25 50.20 0.00 0.00

13:45:36 32.14 58.48 0.00 0.00

```
13:45:41 79.80 58.40 0.00 0.00
```

```
Average 49.37 55.69 0.00 0.00
```

vflt/s 每秒进行页面故障地址转换的数量（由于有效的页面当前不在内存中）。

pflt/s 每秒来自由于保护错误出现的页面故障数量（由于对页面的非法存，取引起的页面故障）。

pgfil/s 每秒通过”页—入”满足 vflt/s 的数量。

rclm/s 每秒由系统恢复的有效页面的数量。有效页面被增加到自由页面队列上。

★如果 vflt/s 的值高于 100，可能预示着对于页面系统来说，应用程序的效率不高，也可能分页参数需要调整，或者内存配置不太合适。

-q 报告进程队列（运行队列和交换队列的平均长度）情况

```
sar -q 2 3
```

```
SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/13/2002
```

```
14:25:50 runq-sz %runocc swpq-sz %swpocc (-q)
```

```
14:25:52 4.0 50
```

```
14:25:54 9.0 100
```

```
14:25:56 9.0 100
```

```
Average 7.3 100
```

runq-sz 准备运行的进程运行队列。

%runocc 运行队列被占用的时间（百分比）

swpq-sz 要被换出的进程交换队列。

%swpocc 交换队列被占用的时间（百分比）。

★如果%runocc 大于 90，并且 runq-sz 的值大于 2，则表明 CPU 的负载较重。其直接后果，可能使系统的响应速度降低。如果%swpocc 大于 20，表明交换活动频繁，将严重导致系统效率下降。解决的办法是加大内存或减少缓存区数量，从而减少交换及页—入，页—出活动。

-r 报告内存及交换区使用情况（没有使用的内存页面和硬盘块）

```
sar -r 2 3
```

```
SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/14/2002
```

```
10:14:19 freemem freeswp availrmem availsmem (-r)
```

```
10:14:22 279729 6673824 93160 1106876
```

```
10:14:24 279663 6673824 93160 1106876
```

```
10:14:26 279661 6673824 93160 1106873
```

```
Average 279684 6673824 93160 1106875
```

freemem 用户进程可以使用的内存页面数，4KB 为一个页面。

freeswp 用于进程交换可以使用的硬盘盘块，512B 为一个盘块。

-u CPU 利用率

```
sar -u 2 3
```


SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/14/2002

10:27:23 %usr %sys %wio %idle (-u)

10:27:25 2 3 8 88

10:27:27 3 3 5 89

10:27:29 0 0 0 100

Average 2 2 4 92

.

%usr cpu 处在用户模式下时间 (百分比)

%sys cpu 处在系统模式下时间 (百分比)

%wio cpu 等待输入, 输出完成 (时间百分比)

%idle cpu 空闲时间 (百分比)

* 在显示的内容中, %usr 和 %sys 这两个值一般情况下对系统无特别影响, %wio 的值不能太高, 如果 %wio 的值过高, 则 CPU 花在等待输入, 输出上的时间太多, 这意味着硬盘存在 I/O 瓶颈。如果 %idle 的值比较高, 但系统响应并不快, 那么这有可能是 CPU 花时间等待分配内存引起的。%idle 的值可以较深入帮助人们了解系统的性能, 在这种情况下, %idle 的值处于 40~100 之间, 一旦它持续低于 30, 则表明进程竞争的主要资源不是内存而是 CPU。

* 在有大量用户运行的系统中, 为了减少 CPU 的压力, 应该使用智能多串卡, 而不是非智能多串卡。智能多串卡可以承担 CPU 的某些负担。

* 此外, 如果系统中有大型的作业运行, 应该把它们合理调度, 错开高峰, 当系统相对空闲时再运行。

-v 报告系统表的内容 (进程, i 节点, 文件和锁表状态)

sar -v 2 3

SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/14/2002

10:56:46 proc-sz ov inod-sz ov file-sz ov lock-sz (-v)

10:56:48 449/ 500 0 994/4147 0 1313/2048 0 5/ 128

10:56:50 450/ 500 0 994/4147 0 1314/2048 0 5/ 128

10:56:52 450/ 500 0 994/4147 0 1314/2048 0 5/ 128

proc-sz 目前在核心中正在使用或分配的进程表的表项数

inod-sz 目前在核心中正在使用或分配的 i 节点表的表项数

file-sz 目前在核心中正在使用或分配的文件表的表项数

ov 溢出出现的次数

lock-sz 目前在核心中正在使用或分配的记录加锁的表项数

* 除 ov 外, 均涉及到 unix 的核心参数, 它们分别受核心参数 NPROC, NIMODE, NFILE 和 FLOCKREC 的控制。

* 显示格式为:

实际使用表项/整个表可以使用的表项数

比如, proc-sz 一行所显示的四个数字中, 分母的 100 是系统中整个进程表的长度 (可建立 100 个表项), 分子上的 24, 26 和 25 分别是采样的那一段时间所使用的进程表项。inod-sz, file-sz 和 lock-sz 三列数字的意义也相同。

三列 ov 的值分别对应进程表, i 节点表和文件表, 表明目前这三个表都没有出现溢出现象, 当出现溢出时, 需要调整相应的核心参数, 将对应表加大。

-w 系统交换活动

```
sar -w 2 3
```

```
SCO_SV scosvr 3.2v5.0.5 PentII(D)ISA 06/14/2002
```

```
11:22:05 swpin/s bswin/s swpot/s bswots pswch/s (-w)
```

```
11:22:07 0.00 0.0 0.00 0.0 330
```

```
11:22:09 0.00 0.0 0.00 0.0 892
```

```
11:22:11 0.00 0.0 0.00 0.0 1053
```

```
Average 0.00 0.0 0.00 0.0 757
```

swpin/s 每秒从硬盘交换区传送进入内存的次数。

bswin/s 每秒为换入而传送的块数。

swpot/s 每秒从内存传送到硬盘交换区的次数。

bswots 每秒为换出而传送的块数。

pswch/s 每秒进程交换的数量。

*swpin/s, bswin/s, swpot/s 和 bswots 描述的是与硬盘交换区相关的交换活动。交换关系到系统的效率。交换区在硬盘上对硬盘的读、写操作比内存读、写慢得多，因此，为了提高系统效率就应该设法减少交换。通常的作法就是加大内存，使交换区中进行的交换活动为零，或接近为零。如果 swpot/s 的值大于 1，预示可能需要增加内存或减少缓冲区（减少缓冲区能够释放一部分自由内存空间）。

8、proc 文件系统

“proc 文件系统是一个伪文件系统，它只存在内存当中，而不占用外存空间。它以文件系统的方式为访问系统内核数据的操作提供接口。用户和应用程序可以通过 proc 得到系统的信息，并可以改变内核的某些参数。”

这里将介绍如何从 /proc 文件系统中获取与防火墙相关的一些性能参数，以及如何通过 /proc 文件系统修改内核的相关配置。

1、从 /proc 文件系统获取相关的性能参数

cpu 使用率：/proc/stat

内存使用情况：/proc/meminfo

网络负载信息：/proc/net/dev

相应的计算方法：（摘自：什么是 proc 文件系统）

- (1) 处理器使用率
- (2) 内存使用率
- (3) 流入流出数据包
- (4) 整体网络负载

这些数据分别要从 /proc/stat、/proc/net/dev、/proc/meminfo 三个文件中提取。如里有问题或对要提取的数据不太清楚，可以使用 man proc 来查看 proc 文件系统的联机手册。

(1) 处理器使用率

这里要从 /proc/stat 中提取四个数据：用户模式(user)、低优先级的用户模式(nice)、内核模式(system)以及空闲的处理器时间(idle)。它们均位于 /proc/stat 文件的第一行。CPU 的利用率使用如下公式来计算。

CPU 利用率 = $100 * (\text{user} + \text{nice} + \text{system}) / (\text{user} + \text{nice} + \text{system} + \text{idle})$

(2) 内存使用率

这里需要从/proc/meminfo 文件中提取两个数据，当前内存的使用量(cmem)以及内存总量(amem)。

内存使用百分比 = 100 * (cmem / umem)

(3) 网络利用率

为了得到网络利用率的相关数据，需要从/proc/net/dev 文件中获得两个数据：从本机输出的数据包数，流入本机的数据包数。它们都位于这个文件的第四行。

性能收集程序开始记录下这两个数据的初始值，以后每次获得这个值后均减去这个初始值即为从集群启动开始从本节点通过的数据包。

利用上述数据计算出网络的平均负载，方法如下：

平均网络负载 = (输出的数据包+流入的数据包) / 2

2. 通过/proc 文件系统调整相关的内核配置

允许 ip 转发 /proc/sys/net/ipv4/ip_forward

禁止 ping/proc/sys/net/ipv4/icmp_echo_ignore_all

可以在命令行下直接往上述两个“文件”里头写入“1”来实现相关配置，如果写入“0”将取消相关配置。

不过在系统重启以后，这些配置将恢复默认设置，所以，如果想让这些修改生效，可以把下面的配置直接写入/etc/profile 文件，或者其他随系统启动而执行的程序文件中。

1.echo 1 > /proc/sys/net/ipv4/ip_forward

2.echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all

如果需要获取其他的性能参数，或者需要对内核进行更多的配置，可以参考 proc 文件系统介绍，也可以直接通过 man proc 查看相关的信息。

十二、 系统服务

1、系统服务分类，根据其使用的方法来分，可以被分为三类

a、由 init 控制的服务：基本都是系统级别的服务，运行级别这一章讲的就是这一类的服务

b、由 System V 启动脚本启动的服务：和我们打交道最多的一种服务，服务器基本都是这个类型的服务

c、由 xinetd 管理的服务

2、System V启动脚本启动的服务

/etc/rc.d/init.d/目录下的内容如下：这些常用的服务器都是 System v 的服务，要控制 System V 的服务，我们可以使用以下命令

#/etc/rc.d/init.d/script {start|stop|restart|reload|condrestart|status}

stop：停止这个服务。

restart：先停止，再启动，也就是重新启动的意思。

reload：重新加载设定档，这个参数只有在服务已经启动的状况下才能使用。

condrestart：有条件的重新启动，这个服务必须是已经启动的，才会被重新启动；如果这个服务尚未启动，则无须启动之。

status：察看目前服务的启动状态。

也可以使用 service 命令来执行脚本，例如 #service network

{start|stop|restart|reload|condrestart|status}

```
[root@test init.d]# ls
acpid          dc_server      iscsi          netconsole     readahead_later  vmware-tools
anacron        dnsmasq        iscsid         netfs          restorecond      vncserver
apmd           dund           isdn           netplugd       rhnsd            wdaemon
atd            firstboot      kdump          network        rpcgssd          winbind
auditd         functions      killall        NetworkManager rpcidmapd        wpa_supplicant
autofs         gpm            krb524         nfs            rpcsvcgssd       xend
avahi-daemon   haldaemon      kudzu          nfslock        saslauthd        xenddomains
avahi-dnscfnd  halt           libvirt        nsd            sendmail         xfs
bluetooth      hidd           lvm2-monitor   ntpd           setroubleshoot   xinetd
capi           hplip          mcstrans       pand           single           ypbind
conman         httpd          mdmonitor      pcsd           smartd           yum-updatesd
cpuspeed       ip6tables      mdmmpd         portmap        smb
crond          ipmi           messagebus     psacct         squid
cups           iptables       microcode_ctl  rawdevices     sshd
cups-config-daemon irda          multipathd      rdisc          syslog
dc_client      irqbalance     named           readahead_early tux
```

System V 的服务在不同级别下的默认开关可以不相同。我们还可以用两种方法来控制默认情况下，开机是否开启某些服务，使用 chkconfig 和 ntsysv（图形方式，默认只能定义当前级别，不过可以增加参数来实现如 # ntsysv -level 23）来控制。

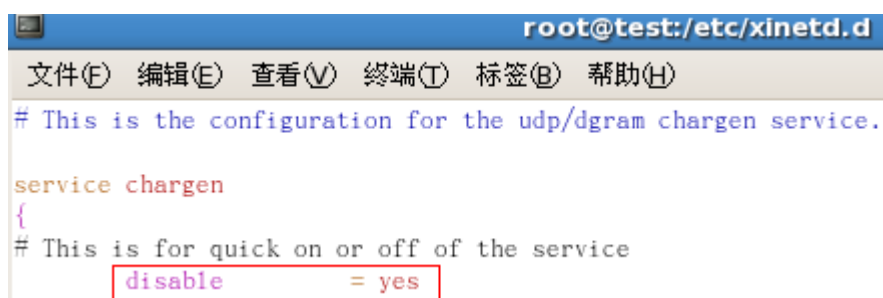
```
#chkconfig --list           //查看系统 system v 服务所有级别下的开关情况。
#chkconfig sshd on|off      //更改 sshd 服务 2-5 级别的默认开关情况
#chkconfig sendmail off     //所有级别关闭 sendmail 服务
#chkconfig --level 23 sendmail off //在 2、3 级别关闭 sendmail 服务
```

3、xinetd服务管理

xinetd 服务的管理文件都放在 /etc/xinetd.d 目录内，我们可以编辑这个目录内的服务文件来开启和关闭服务。每个服务文件都有 disable 这个行，如果把值改成 yes 就是禁用服务，如果是 no，那就是启动这个服务。修改成功后，要使修改生效，需要重新启动 xinetd 服务。

```
#service xinetd restart
```

```
[root@test xinetd.d]# ls
chargen-dgram  daytime-stream  echo-stream  klogin  rsync
chargen-stream  discard-dgram  eklogin      krb5-telnet  tcpmux-server
cvs            discard-stream  ekrb5-telnet  kshell  tftp
daytime-dgram  echo-dgram      gssftp       rmcp       time-dgram
# vim chargen-dgram
```



```
root@test:/etc/xinetd.d
文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
# This is the configuration for the udp/dgram chargen service.

service chargen
{
# This is for quick on or off of the service
  disable = yes
```

4、常见服务列表

服务名	必需（是/否）	用途描述	注解
acon	否	语言支持	特别支持左手书写语言:阿拉伯语, 波斯语和希伯来语
acpi	否	电源管理	手提电脑电池电扇监控器
acpid	否	监听精灵进程	此进程监听并分配内核中的 acpi 事件

adsl	否	内部 ADSL 开关控制	只有你的计算机内部有互联网连接	adsl
开关时才用到此服务				
alsa	否	高级 Linux 声音构件	这个单独的声音系统实际包含在内核中	
anacron	否	周期命令调度程序	一个任务调度工具	
apmd	否	电源管理	手提电脑电源管理	
apmiser	否	电源管理	另一手提电脑电池延长器	
arpwatch	否	以太网 IP 地址配对监控器	用主机名监控并记录远程 IP 地址	
atd	否	周期命令调度程序	一个任务调度工具	
autofs	否	自动安装服务	几个命令服务文件系统自动安装之一. 一些此类服务专门针对发行配套软件, 如果你使用的发行配套软件拥有自己的自动安装系统, 不要用这一个.	
bluetooth	否	蓝牙技术核心	用于所有蓝牙服务	
bootparamd	否	导入服务	以前导入无盘客户端/瘦客户端的方法. 最新型的方法为零配置系统(zeroconf system).	
canna	否	日语转换引擎		
capi4linux	否	基本 CAPI 子系统		
cpqarrayd	否	硬件服务	康柏独立冗余磁盘阵列(Raid Array)监控器	
cpufreq	否	硬件服务	检查并配置 CPU 频率精灵程序模块	
cpufreqd	否	硬件服务	此服务自动衡量 CPU 频率来减少过热情况. 在超频时有用.	
crond	是	周期命令调度程序	一个任务调度工具	
Cups-lpd	否	使旧式 Linux 或商业 Unix 系统连接到打印主机上.	只有在允许旧式系统访问打印机时才有用	
cups	是	公共 Unix 打印系统	进行打印的必要功能	
cvs	否	并发版本系统	用于管理多用户文档	
devfsd	否	系统维护	此服务只清除动态桌面目录, 除非你的系统经常崩溃, 否则不需要此服务.	
dhcpcd	否	DHCP 服务器	你的网络足够大, 使用静态 IP 很麻烦吗? 此项服务对你的网络进行 DHCP IP 配置, 方便网络应用.	
diald	否	拨号网络智能自动拨号器	此服务一经请求, 即连接上网络. 你一旦输入电子邮件, 点击发送, 它就自动连接, 发送电邮并断开.	
dkms	否	DKMS 自安装导入	发行配套软件专用工具, 用于 OEM 类型安装. 它允许管理员密码的最初导入设置以及常规应用的用户名密码, 系统的最后配置.	
dm	是	显示管理器	X 服务器的核心, 使用图形用户界面(GUI)时必需.	
dnbc	否	数字网络绑定 Chrooter	这是一个简单的 bash 脚本, 它将一个 BIND 服务器放入一个 chroot 牢笼中. 安装 BIND, 发布脚本并重启.	
Drakxtools-http	否	小型服务管理服务器	远程系统管理的发行配套软件专用工具.	
dund	否	蓝牙拨号网络		
fam	否	文件系统变更监控器	文件系统所有改变的记录器	
finger	否	数据远程访问	此服务允许你远程访问用户登录日期, 最后登录日期与时间. 用于不在办公室时监控雇员的工作习惯, 主要的安全违反, 因为你正有效地在线发布公司机密数据.	
freshclam	是	ClamAV 更新器	用于自动更新 ClamAV	

gpm	是	鼠标	鼠标驱动器控制台模式
haldaemon	否	硬件监控系统	此服务监控硬件改变, 为你改变新的或更改过的硬件.
harddrake	否	硬件服务	发行配套软件专用硬件探测与配置
heartbeat	否	高可用性服务	此服务旨在增加重要服务与服务器的优先级
hidd	否	蓝牙 H. I. D. 服务器	
hplip	否	惠普 Linux 打印与成像	旧版惠普集成产品供应驱动器
hpoj	否	Pital?init, 惠普办公喷墨打印机驱动器	惠普办公喷墨打印机旧式驱动器. 新式驱动器包含在打印机的打印驱动器内.
httpd	否	Apache 网络服务器	在系统上应用此服务有两个原因, 一是要用它作为网络服务器, 二是用它作为网址开发器. 如果没有此二项, 则不必安装 Apache.
hylafax?server	否	企业传真机?调制调解器服务	此服务仅用于 1 类与 2 类传真机. 如果你想用 hylafax 通过调制调解器发送传真, 必须运行此服务. 它并不是唯一有效的传真工具.
ibod	否	按需 ISDN MPPP 带宽	与拨号网络一同使用, 按需连接到网络.
identd	否	TCP 连接鉴定	
imaps	否	安全 IMAP 服务器	
imaps	否	IMAP 服务器	
iplog	否	用主机名或远程主机记录 TCP, UDP, ICMP.	有用的网络监控工具
ipop2	否	POP2 邮件服务器	
ipop3	否	POP3 邮件服务器	
ipsec	否	加密与验证通信	KLIPS 为内核一半, PLUTO 为用户空间一半. 在远程访问情况下十分有用.
iptables	是	基于 Packet 过滤防火墙内核	所有优秀的 Linux 防火墙都基于此项服务
ipvsadmin	否	Linux 核心 IP 虚拟服务器	最早的 Linux 网络系统之一, 已不常用.
irda	否	红外线设备界面	以前的无线设备支持
keytable	是	键盘映射	此服务明确告诉系统你正在使用哪种键盘
kheader	否	导入服务	此服务自动重建内核头导入
lads	否	登录异常探测系统	跟踪登录企图并警告入侵企图的工具
laptop mode	否	电源管理	减少电力耗费, 延长手提电脑电池寿命的工具
leafnode	否	X? INETD NNTP 服务	
lisa	否	局域网信息服务器	
lmsensors	否	硬件健康监控器	此服务要求系统主板支持并有合适的监控系统, 如 CPU 温度与电压监控器.
mailman	否	GUN 邮件列表管理器	常用的邮件列表工具, 带 Python 编写的管理网络界面. 它允许列表成员发送邮件并回复邮件到同一个地址进行交流. 它还可用于向那些发送请求的用户传送新闻时讯/产品更新.
mandi	否	交互式防火墙	允许暂时无线访问系统的专用服务, 将为当前任务开放 iptables 防火墙, 仅用于无线设备访问. 在用户许可情况下才可使用, 不能自动使用.
mdadm	否	软阵列监控服务	这也是一个用于上述软件阵列栏的管理工具
mdnsresponder	否	零配置 DNS 配置工具	

messagebus	是	事件监控服务	此服务在必要时向所有用户发送广播信息, 如服务器将要重启.
mon	是	系统监控精灵进程	许多系统服务要求此服务来运行
mysqld	否	MySQL 服务器	如果你不需要这个数据库, 不要打开它.
named	否	绑定 (BIND) 服务器	这就是声名狼藉的名称服务器
netplugd	否	网卡精灵进程	此服务监控网络界面, 根据信号关闭或启动它, 主要用于不经常连接的手提电脑.
network	是	网络	此服务打开网卡, 或为调制解调器供电.
nfs	否	网络文件共享	此服务使用户访问 NFS 共享文件, 为 NFS 系统客户机所必需.
nfsfs	否	网络文件共享服务器	只有在网络服务器上才被激活
nfslock	否	NFS 文件锁定	只有在使用 NFS 网络/文件共享功能时, 此服务才被激活.
nifd	否	Howl 客户端	此服务为零配置网络/系统提供 ipv4 链接本地服务
nsd	否	密码与群查找服务	此服务用于减慢 N. I. S/Y. P. nist, ldap 和 hesiod 之类的服务. 专门为这些服务提供更长的中断时间.
ntpd	否	NTP 服务器的第 4 版	
numlock	否	数字锁定键灯光控制	此服务保持数字锁定键的激活状态, 打开键盘上的数字键区.
Okidataemon	否	OKI4 和兼容 win 打印机的兼容性精灵进程	只有在你有这些打印机时才可用
pand	否	蓝牙个人区域网络	用于基于网络的家庭区域蓝牙技术
partmon	是	分区监控	此服务跟踪安装分区上的剩余空间. 大多数文件系统浏览器使用它来计算指定分区上的剩余空间.
pcmcia	否	个人电脑内存卡国际协会	
pg_autovacuum	否	PostgreSQL 维护	此服务自动运行 PostgreSQL 所需的空间 (vacuum) 来减少磁盘空间, 从数据库中拖动临时表格, 并删除 PostgreSQL 建立的临时文件.
pop3s	否	安全 POP3 服务	POP3 SSL 服务器
portmap	否	RPC 支持	支持那些应用 rpc 的罕有的应用软件
postgresql	否	Postgresql 数据库引擎	只有在运行或开发 Postgresql 数据库驱动应用软件时才用到此服务
postfix	否	电子邮件服务器	与 sendmail 兼容的电子邮件服务器, 比 sendmail 更新, 也变得比 sendmail 更通用.
pptp	否	PPP 断电服务	PPP 频道断电服务, UPS 打开时使用, 以避免电源返回系统时出现文件锁定问题.
prelude	否	IDS	入侵探测系统
psacct	否	进程计算	活动进程追踪器, 实际上是资源的浪费.
rawdevices	是	分配 raw 设备, 阻止其使用	DVDS, oracle DBMS 等需要此服务
rsync	否	远程同步	使指定目录树上的文件远程同步的服务器, 常用于维护镜像地址, 也在备份时用于保持公司文件为最新状态.
saned	否	网络扫描仪	从网络上的任何工作站提供扫描仪访问
shorwall	是	防火墙	一个非常优秀的 IPTables 防火墙

smartd	否	自我监控服务	用于智能设备的 OS 访问, 此服务允许 Linux 告诉你是否设备将要变坏, 但这要依靠设备的精确智能特性.
smb	否	Samba 网络服务	此服务提供 samba 服务, 实现 Windows 网络兼容性.
snmpd	否	简单的网络管理协议	用于小型(家庭办公室)网络
sound	否	声音系统	此为 Linux 声音系统的核心, 适用于桌面系统, 在服务器上则是资源的浪费.
squid	否	高速缓存工具	用于高速缓存网络页面及 DNS 登录
ssh?xinetd	否	X?inetd OpenSSH 服务器	OpenSSH 的按需运行版本
sshd	否	OpenSSH 服务器	如果你需要 SSH 访问你的系统时才开启此服务, 将不会使用 x?inetd 版本.
subversion	否	并发版本系统	CVS 的新型替代品
swat	否	Samba 网络管理工具	基于 Samba 管理的网络
syslog	是	系统登录	一项必要的服务, 控制整个系统上的所有登录.
tmdns	否	多点传送 DNS 响应器	用于零配置环境
ultrabayd	否	ThinkPad 工具	此服务为你的 IBM ThindPad 探测 ultrabay, 并在适当情况下启动/关闭 IDE 接口.
upsd	否	NUT 精灵进程及驱动器	一个不间断地电源监控及报告工具, 此服务向中心地址报告, 产生关于 UPS 统计的数据库.
upsmon	否	UPS 监控工具	此服务监控 UPS 的状况, 在其运行低下时关系系统.
vncserver	否	虚拟网络计算服务器	在项目中应用 VNC 时非常有用
Webmin	否	远程管理工具	发行配套软件 Agnostic 远程管理工具. 在机器不能总是直接访问, 如网络服务器集群时有用.
winbind	否	Samba 名称服务器	Samba 网络运行所必需. 此服务将用户与群数据从 windows 网络映射到 Linux 工作站中.
wine	否	Wine 并非竞争者	此服务使 MS Windows 可在 Linux 上执行, WINE 是商业产品 Crossover Office 的限制版本.
wlan	否	控制精灵进程	由于服务通常由 init 进程控制, 此控制服务不常用.
xinetd	是	监控并控制其它服务器的服务器	这是一项必需的服务, 它实际上减少了服务器上 CPU 的负载. 如果你需要 SSH, ftp 等但并不总是需要, x?inetd 版本将在请求, 甚至是远程需求时启动它们. 此服务让它们生效, 但如果它们一天/周只使用几次的话, 又释放了时钟周期.
xfs	是	X 字体服务器	你任何时间需要使用图形用户界面(GUI), 就需要此服务.
ypbind	否	SUN 的 YP 服务器名称服务器	此服务用于基于 GLIBC 的 NIS/YP 网络服务

十三、 环境管理

1、环境变量

在 linux 系统下，如果你下载并安装了应用程序，很有可能在键入它的名称时出现“command not found”的提示内容。如果每次都到安装目标文件夹内，找到可执行文件来进行操作就太繁琐了。这涉及到环境变量 PATH 的设置问题，而 PATH 的设置也是在 linux 下定制环境变量的一个组成部分。

环境变量可以让子程序继续引用的原因，是因为：

1. 当启动一个 shell，操作系统分配一记忆区块给 shell 使用，此区域之变量可以让子程序存取；

2. 利用 export 功能，可以让变量的内容写到上述的记忆区块当中(环境变量)；

当加载另一个 shell 时（亦即启动子程序，而离开原本的父程序了），子 shell 3. 可以将父 shell 的环境变量所在的记忆区块导入自己的环境变量区块当中。

所以环境变量是和 Shell 紧密相关的，用户登录系统后就启动了一个 Shell。对于 Linux 来说一般是 bash，但也可以重新设定或切换到其它的 Shell（使用 chsh 命令）。

根据发行版本的情况，bash 有两个基本的系统级配置文件：/etc/bashrc 和/etc/profile。这些配置文件包含两组不同的变量：shell 变量和环境变量。前者只是在特定的 shell 中固定（如 bash），后者在不同 shell 中固定。很明显，shell 变量是局部的，而环境变量是全局的。环境变量是通过 Shell 命令来设置的，设置好的环境变量又可以被所有当前用户所运行的程序所使用。对于 bash 这个 Shell 程序来说，可以通过变量名来访问相应的环境变量，通过 export 来设置环境变量。

注：Linux 的环境变量名称一般使用大写字母

1.1、 Linux环境变量的种类

按环境变量的生存周期来划分，Linux 的环境变量可分为两类：

①永久的：需要修改配置文件，变量永久生效。

②临时的：使用export命令行声明即可，变量在关闭shell时失效。

1.2、设置变量的三种方法

①在/etc/profile文件中添加变量对所有用户生效（永久的）

用 VI 在文件/etc/profile 文件中增加变量，该变量将会对 Linux 下所有用户有效，并且是“永久生效”。

例如：编辑/etc/profile 文件，添加 CLASSPATH 变量

```
# vi /etc/profile
export CLASSPATH=../JAVA_HOME/lib;JAVA_HOME/jre/lib
```

注 1：profile 文件在系统启动时将被运行。大家可以在里面加入其他命令，但是一定要加正确，不然的话系统会启动不起来的。

②在用户目录下的.bash_profile文件中增加变量对单一用户生效（永久的）

用 VI 在用户目录下的.bash_profile 文件中增加变量，改变量仅会对当前用户有效，并且是“永久的”。

例如：编辑 guok 用户目录（/home/guok）下的.bash_profile

```
# vi /home/guok/.bash.profile
```

添加如下内容：

```
export CLASSPATH=./JAVA_HOME/lib:$JAVA_HOME/jre/lib
```

注 2: 如果修改了/etc/profile, 那么编辑结束后执行 source profile 或 执行点命令 ./profile, PATH 的值就会立即生效了。这个方法的原理就是再执行一次/etc/profile shell 脚本, 注意如果用 sh /etc/profile 是不行的, 因为 sh 是在子 shell 进程中执行的, 即使 PATH 改变了也不会反应到当前环境中, 但是 source 是在当前 shell 进程中执行的, 所以我们能看到 PATH 的改变。

注 3: 变量重复定义时, 以后面的设置为先。

例如: 在 peofile 文件默认对 PATH 变量都有设置 PATH=¥¥¥¥¥¥¥¥, 在以后可能在对 PATH 设置, 一般都加在 profile 文件的最后 PATH= (打个比方)。而系统之中认定的 PATH= ¥¥¥¥¥¥¥¥¥¥, 也就是说相同名字的环境变量, 后写入的先起作用 (通俗地讲)。

注 4、特殊字符介绍。

例如在 profile 中有如下内容, 通过以下内容说明特殊符号的用法。

```
export A=/q/jing:aaa/cc/ld
```

```
export B=./liheng/wang export A=/cd/cdr:$A
```

: 表示并列含义, 例如 A 变量值有多个, 用: 符号进行分离。

. 表示你操作的当前目录。例如 pap 命令会查找 B 环境变量。

在/home 键入 B 命令, 系统首先在/home 目录下 (即当前路径) 查找关于 B 的内容, 如果没有在 /liheng/wang 目录下查找关于 B 的内容。\$ 表示该变量本次定义之前的值, 例如 \$A 代表 /q/jing:aaa/cc/ld。也就是说 A=/cd/cdr:/q/jing:aaa/cc/ld

注 5、常见的环境变量

PATH: 决定了 shell 将到哪些目录中寻找命令或程序

HOME: 当前用户主目录

MAIL: 是指当前用户的邮件存放目录。

SHELL: 是指当前用户用的是哪种 Shell。

HISTSIZE: 是指保存历史命令记录的条数。

LOGNAME: 是指当前用户的登录名。

HOSTNAME: 是指主机的名称, 许多应用程序如果要用到主机名的话, 通常是从这个环境变量中取得的。

LANG/LANGUGE: 是和语言相关的环境变量, 使用多种语言的用户可以修改此环境变量。

PS1: 是基本提示符, 对于 root 用户是#, 对于普通用户是\$。

PS2: 是附属提示符, 默认是“>”。可以通过修改此环境变量来修改当前的命令符, 比如下列命令会将提示符修改成字符串“Hello, My NewPrompt :)”。

```
# PS1=" Hello, My NewPrompt :) "
```

③直接运行export命令定义变量【只对当前shell (BASH) 有效 (临时的)】

在 shell 的命令行下直接使用[export 变量名=变量值]定义变量, 该变量只在当前的 shell (BASH) 或其子 shell (BASH) 下是有效的, shell 关闭了, 变量也就失效了, 再打开新 shell 时就没有这个变量, 需要使用的话还需要重新定义。

1.3、环境变量设置命令

①echo \$ <变量名> //显示某个环境变量

②env // environment (环境) 的简写, 列出来所有的环境变量

③set //显示所有本地定义的Shell变量，这个命令除了会将环境变量列出来之外，其它我们的自定义的变量，都会被列出来。因此，想要观察目前 shell 环境下的所有变量，就用 set 即可！

④export 命令

功能说明：设置或显示环境变量。

语 法：export [-fnp][变量名称]=[变量设置值]

补充说明：在 shell 中执行程序时，shell 会提供一组环境变量。export 可新增，修改或删除环境变量，供后续执行的程序使用。export 的效力仅及于该此登陆操作。

参 数：

-f 代表[变量名称]中为函数名称。

-n 删除指定的变量。变量实际上并未删除，只是不会输出到后续指令的执行环境中。

-p 列出所有的 shell 赋予程序的环境变量。

一个变量创建时，它不会自动地为在它之后创建的 shell 进程所知。而命令 export 可以向后面的 shell 传递变量的值。当一个 shell 脚本调用并执行时，它不会自动得到原为脚本（调用者）里定义的变量的访问权，除非这些变量已经被显式地设置为可用。export 命令可以用于传递一个或多个变量的值到任何后继脚本。

⑤unset清除环境变量，如果未指定值，则该变量值将被设为NULL

⑥readonly设置只读变量，只读变量设置后不能用unset清除，除非重启shell

⑦declare 、typeset 这两个命令是完全一样的，他们允许指定变量的具体类型，在某些特定的语言中，这是一种指定类型的很弱的形式，declare 命令是在 Bash 版本 2 或之后的版本才被加入的，typeset 命令也可以工作在 ksh 脚本中。

```
[root@linux ~]# declare [-aixr] variable
```

参数：

-a : 将后面的 variable 定义成为数组 (array)

-i : 将后面接的 variable 定义成为整数数字 (integer)

-x : 用法与 export 一样，就是将后面的 variable 变成环境变量；

-r : 将一个 variable 的变量设定成为 readonly，该变量不可被更改内容，也不能 unset

范例一：让变量 sum 进行 100+300+50 的加总结果

```
[root@linux ~]# sum=100+300+50
```

```
[root@linux ~]# echo $sum
```

100+300+50 <==咦！怎么没有帮我计算加总？因为这是文字型态的变量属性啊！

```
[root@linux ~]# declare -i sum=100+300+50
```

```
[root@linux ~]# echo $sum
```

450

范例二：将 sum 变成环境变量

```
[root@linux ~]# declare -x sum
```

范例三：让 sum 变成只读属性，不可更动！

```
[root@linux ~]# declare -r sum
```

```
[root@linux ~]# sum=tesgting
```

-bash: sum: readonly variable

十四、 网络管理

网卡在 Linux 操作系统中用 ethX, 是由 0 开始的正整数, 比如 eth0、eth1..... ethX。而普通猫和 ADSL 的接口是 pppX, 比如 ppp0 等

7.1、ifconfig

1、 关于网络接口及配置工具说明;

在 Linux 操作系统中配置网络接口, 一般是通过网络配置工具实现的, 但最终目的还是通过网络配置工具来达到修改与网络相关的配置文件而起作用的。由此说来, 我们配置网络可以直接修改配置文件。

比如网络接口 (网卡) 的 IP 地址、子掩码、网关, 在 Slackware 中只需修改一个配置文件就行了 /etc/rc.d/rc.inet1, 而在 Redhat/Fedora 等或以 Redhat/Fedora 为基础的发行版中, 一般要涉及到好几个文件, 比如包括 /etc/sysconfig/network-scripts/ifcfg-eth0 在内等。

了解 Linux 网络配置文件是极为重要的, 我们通过工具修改了什么, 是怎么生效的, 只有了解网络配置文件才能搞清楚。做个不恰当的比喻: Linux 系统是一个透明的盒子, 至于盒子里装的是什么都是一目了然的。而闭源操作系统, 我们没有机会知道这些, 更不知道他是怎么实现的。

对于复杂的网络模型, Linux 操作系统是有极大的优势, 可能在我们看看 man 和 help, 修改修改配置文件, 在几分钟就可以搞定。但闭源图形界面的操作系统就没有这么幸运了, 反复的点鼠标。

点了几十上百次也解决不了一个问题, 这是极为常见的。由于 Linux 操作系统存在很多的发行和版本, 大多发行版本都有自己的专用配置工具。主要是为了方便用户配置网络; 但也有通用的配置工具, 比如 Linux ifconfig、ifup、ifdown;

2 关于网络硬件设备的驱动;

我在以前的文档中有写过, 网络硬件, 比如网卡 (包括有线、无线), 猫包括普通猫和 ADSL 猫等, 都是需要内核支持的, 所以我们首先得知道我们的网络设备是不是已经被硬内核支持了。如果不支持, 我们得找驱动 (或通过内核编译) 来支持它; 请参考:

3、Linux ifconfig 配置网络接口的工具介绍;

Linux ifconfig 是一个用来查看、配置、启用或禁用网络接口的工具, 这个工具极为常用的。比如我们可以用这个工具来临时性的配置网卡的 IP 地址、掩码、广播地址、网关等。也可以把它写入一个文件中 (比如/etc/rc.d/rc.local), 这样系统引导后, 会读取这个文件, 为网卡设置 IP 地址;

不过这样做目前看来没有太大的必要。主要是各个发行版本都有自己的配置工具, 无论如何也能把主机加入到网络中; 下面我们看看 Linux ifconfig 用法;

3.1 Linux ifconfig 查看网络接口状态;

Linux ifconfig 如果不接任何参数, 就会输出当前网络接口的情况;

```
1. [root@localhost ~]# Linux ifconfig
2. eth0      Link encap:Ethernet  HWaddr 00:C0:9F:94:78:0E
3.  inet addr:192.168.1.88  Bcast:192.168.1.255  Mask:255.255.255.0
4.  inet6 addr: fe80::2c0:9fff:fe94:780e/64 Scope:Link
5.  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
6.  RX packets:850 errors:0 dropped:0 overruns:0 frame:0
7.  TX packets:628 errors:0 dropped:0 overruns:0 carrier:0
8.  collisions:0 txqueuelen:1000
9.  RX bytes:369135 (360.4 KiB)  TX bytes:75945 (74.1 KiB)
10. Interrupt:10 Base address:0x3000
11.
12. lo       Link encap:Local Loopback
13.  inet addr:127.0.0.1  Mask:255.0.0.0
14.  inet6 addr: ::1/128 Scope:Host
15.  UP LOOPBACK RUNNING  MTU:16436  Metric:1
16.  RX packets:57 errors:0 dropped:0 overruns:0 frame:0
17.  TX packets:57 errors:0 dropped:0 overruns:0 carrier:0
18.  collisions:0 txqueuelen:0
19.  RX bytes:8121 (7.9 KiB)  TX bytes:8121 (7.9 KiB)
```

解说: eth0 表示第一块网卡, 其中 HWaddr 表示网卡的物理地址, 我们可以看到目前这个网卡的物理地址(MAC 地址) 是 00:C0:9F:94:78:0E ; inet addr 用来表示网卡的 IP 地址, 此网卡的 IP 地址是 192.168.1.88, 广播地址, Bcast:192.168.1.255, 掩码地址 Mask:255.255.255.0

lo 是表示主机的回环地址, 这个一般是用来测试一个网络程序, 但又不想让局域网或外网的用户能够查看, 只能在此台主机上运行和查看所用的网络接口。比如我们把 HTTPD 服务器的指定到回环地址, 在浏览器输入 127.0.0.1 就能看到你所架 WEB 网站了。但只是您能看得到, 局域网的其它主机或用户无从知道;

如果我们想知道主机所有网络接口的情况, 请用下面的命令; [root@localhost ~]# Linux ifconfig -a 如果我们想查看某个端口, 比如我们想查看 eth0 的状态, 就可以用下面的方法; [root@localhost ~]# Linux ifconfig eth0

3.2 Linux ifconfig 配置网络接口;

Linux ifconfig 可以用来配置网络接口的 IP 地址、掩码、网关、物理地址等; 值得一说的是用 Linux ifconfig 为网卡指定 IP 地址, 这只是用来调试网络用的, 并不会更改系统关于网卡的配置文件。

如果您想把网络接口的 IP 地址固定下来，目前有三个方法：一是通过各个发行和版本专用的工具来修改 IP 地址；二是直接修改网络接口的配置文件；三是修改特定的文件，加入 Linux ifconfig 指令来指定网卡的 IP 地址，比如在 redhat 或 Fedora 中，把 Linux ifconfig 的语名写入 /etc/rc.d/rc.local 文件中；

Linux ifconfig 配置网络端口的方法：Linux ifconfig 工具配置网络接口的方法是通过指令的参数来达到目的的，我们只说最常用的参数；Linux ifconfig 网络端口 IP 地址 hw <HW> MAC 地址 netmask 掩码地址 broadcast 广播地址 [up/down]

实例一：

比如我们用 Linux ifconfig 来调试 eth0 网卡的地址

```
1. [root@localhost ~]# Linux ifconfig eth0 down
2. [root@localhost ~]# Linux ifconfig eth0 192.168.1.99 broadcast 192.168.1.255 netmask 255.255.255.0
3. [root@localhost ~]# Linux ifconfig eth0 up
4. [root@localhost ~]# Linux ifconfig eth0
5. eth0      Link encap:Ethernet  HWaddr 00:11:00:00:11:11
6. inet addr:192.168.1.99  Bcast:192.168.1.255  Mask:255.255.255.0
7. UP BROADCAST MULTICAST  MTU:1500  Metric:1
8. RX packets:0 errors:0 dropped:0 overruns:0 frame:0
9. TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
10. collisions:0 txqueuelen:1000
11. RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
12. Interrupt:11 Base address:0x3400
```

注解： 上面的例子我们解说一下；

第一行:Linux ifconfig eth0 down 表示如果 eth0 是激活的,就把它 DOWN 掉。此命令等同于 ifdown eth0;

第二行：用 Linux ifconfig 来配置 eth0 的 IP 地址、广播地址和网络掩码；

第三行：用 Linux ifconfig eth0 up 来激活 eth0 ； 此命令等同于 ifup eth0

第四行：用 Linux ifconfig eth0 来查看 eth0 的状态；

当然您也可以用直接在指令 IP 地址、网络掩码、广播地址的同时，激活网卡；要加 up 参数；比如下面的例子； [root@localhost ~]# Linux ifconfig eth0 192.168.1.99 broadcast 192.168.1.255 netmask 255.255.255.0 up

实例二：在这个例子中，我们要学会设置网络 IP 地址的同时，学会设置网卡的物理地址（MAC 地址）；

比如我们设置网卡 eth1 的 IP 地址、网络掩码、广播地址，物理地址并且激活它；
[root@localhost ~]# Linux ifconfig eth1 192.168.1.252 hw ether 00:11:00:00:11:11 netmask 255.255.255.0 broadcast 192.168.1.255 up 或 [root@localhost ~]# Linux ifconfig eth1 hw ether 00:11:00:00:11:22 [root@localhost ~]# Linux ifconfig eth1 192.168.1.252 netmask 255.255.255.0 broadcast 192.168.1.255 up

其中 hw 后面所接的是网络接口类型， ether 表示以太网， 同时也支持 ax25 、 ARCnet、 netrom 等，详情请查看 man Linux ifconfig ；

3.3 如何用 Linux ifconfig 来配置虚拟网络接口；

有时我们为了满足不同的需要还需要配置虚拟网络接口，比如我们用不同的 IP 地址来架运行多个 HTTPD 服务器，就要用到虚拟地址；这样就省却了同一个 IP 地址，如果开设两个的 HTTPD 服务器时，要指定端口号。

虚拟网络接口指的是为一个网络接口指定多个 IP 地址，虚拟接口是这样的 eth0:0 、 eth0:1、 eth0:2 eth1N。当然您为 eth1 指定多个 IP 地址，也就是 eth1:0、 eth1:1、 eth1:2 以此类推；

其实用 Linux ifconfig 为一个网卡配置多个 IP 地址，就用前面我们所说的 Linux ifconfig 的用法，这个比较简单；看下面的例子；
[root@localhost ~]# Linux ifconfig eth1:0 192.168.1.251 hw ether 00:11:00:00:11:33 netmask 255.255.255.0 broadcast 192.168.1.255 up 或
[root@localhost ~]# Linux ifconfig eth1 hw ether 00:11:00:00:11:33 [root@localhost ~]# Linux ifconfig eth1 192.168.1.251 netmask 255.255.255.0 broadcast 192.168.1.255 up

注意：指定时，要为每个虚拟网卡指定不同的物理地址；

在 Redhat/Fedora 或与 Redhat/Fedora 类似的系统，您可以把配置网络 IP 地址、广播地址、掩码地址、物理地址以及激活网络接口同时放在一个句子中，写入/etc/rc.d/rc.local 中。比如下面的例子；

```
Linux ifconfig eth1:0 192.168.1.250 hw ether 00:11:00:00:11:44 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

```
Linux ifconfig eth1:1 192.168.1.249 hw ether 00:11:00:00:11:55 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

解说：上面是为 eth1 的网络接口，设置了两个虚拟接口；每个接口都有自己的物理地址、IP 地址... ..

3.4 如何用 Linux ifconfig 来激活和终止网络接口的连接；

激活和终止网络接口的用 Linux ifconfig 命令，后面接网络接口，然后加上 down 或 up 参数，就可以禁止或激活相应的网络接口了。当然也可以用专用工具 ifup 和 ifdown 工具；

1. [root@localhost ~]# Linux ifconfig eth0 down
2. [root@localhost ~]# Linux ifconfig eth0 up
3. [root@localhost ~]# ifup eth0
4. [root@localhost ~]# ifdown eth0

对于激活其它类型的网络接口也是如此，比如 ppp0, wlan0 等；不过只是对指定 IP 的网卡有效。注意：对 DHCP 自动分配的 IP，还得由各个发行版自带的网络工具来激活；当然得安装 dhcp 客户端；这个您应该明白；比如 Redhat/Fedora [root@localhost ~]# /etc/init.d/network start
Slackware 发行版： [root@localhost ~]# /etc/rc.d/rc.inet1

4、Debian、Slackware、Redhat/Fedora、SuSE 等发行版专用网络接口配置工具；

由于 Linux ifconfig 用起来比较麻烦，而且是用来测试网络之用，但这个工具并不能修改网络接口的相应配置文件。虽然也能通过把 Linux ifconfig 配置网络接口的语句写入类似/etc/rc.d/rc.local 文件中，但相对来说还是写入关于网络接口的配置文件中更为安全和可靠；但对于虚拟网络接口写入类似/etc/rc.d/rc.local 中还是可以的；

下面我们介绍一下各个发行版的网络接口配置工具；

4.1 Debian 网络接口配置文件和专用配置工具；

正在增加中；

4.2 Redhat/Fedora 网络接口的配置文件和网络接口专用配置工具；

在 Redhat/Fedora 中，与以太网卡相关的配置文件位于 /etc/sysconfig/network-scripts 目录中，比如 ifcfg-eth0、ifcfg-eth1

4.21 Redhat/Fedora 或类似这样的系统，网卡的配置文件；

</FONT?< p>

比如在 Fedora 5.0 中，ifcfg-eth0 ； 如果您用 DHCP 服务器来自动获取 IP 的，一般情况下 ifcfg-eth0 的内容是类似下面这样的；

1. **DEVICE**=eth0
2. **ONBOOT**=yes
3. **BOOTPROTO**=dhcp
4. **TYPE**=Ethernet

如果您是指定 IP 的，一般内容是类似下面的：

1. **DEVICE=eth0** 注：网络接口
2. **ONBOOT=yes** 注：开机引导时激活
3. **BOOTPROTO=static** 注：采用静态 IP 地址；
4. **IPADDR=192.168.1.238** 注：IP 地址
5. **NETMASK=255.255.255.0** 注：网络掩码；
6. **GATEWAY=192.168.1.1** 注：网关；

下面的几个选项也可以利用；

1. **HOSTNAME=linxsir03** 注：指定主机名；
2. **DOMAIN=localdomain** 注：指定域名；
3. **HWADDR=00:00:11:22:00:aa** 注：

指定网卡硬件地址（MAC 地址），也可以省略，不过这在这里来更改 MAC 地址一般是不能生效的。还是通过前面所说的 Linux ifconfig 的办法来更改吧；

4.22 Redhat/Fedora 或类似系统，配置网络的工具介绍；

在 Redhat 早期的版本中，有 linuxconf、redhat-config-network、netconfig 等工具；在 Redhat/Fedora 最新的版本有 system-config-network-tui（文本模式的）、system-config-network（图形模式的），netconfig（文本模式的）。

这些工具都会直接修改 Linux 系统中关于网络接口的配置文件；这是 Linux ifconfig 所不能比的；其中 redhat-config-network 和 system-config-network 工具不仅仅是配置网卡的工具，还有配置 ISDN 和普通猫、ADSL 的工具、网络硬件的添加、主机名字的配置、DNS 各客户端的配置等。其实是一个工具组的集成；

这些工具比较简单，以 root 权限运行命令就能调用，比如：

1. [root@localhost ~]# /usr/sbin/system-config-network
2. [root@localhost ~]# system-config-network

如果您设置了可执行命令的环境变量，不用加路径就可以运行，但前提是您得安装这个网络管理工具；不过值得一说的是 netconfig 工具是一个在文本模式比较好的工具，推荐大家使用；理由是这个工具在文本模式下，也有一个简单的图形界面；还有命令模式；功能强着呢；

1. [root@localhost ~]# netconfig -d eth0 注：配置 eth0
2. [root@localhost ~]# netconfig -d eth1 注：配置 eth1

4.23 Redhat/Fedora 系统中的 netconfig 特别介绍；

netconfig 这个工具，在 Redhat/Fedora 或类似于它们的系统中都是存在的，这个工具比较强大。所以特别介绍一下。但在 Slackware 中 netconfig 是 TEXT 模式下有一个图形模式，但不能象 Linux ifconfig 一样用命令来操作网卡接口；

netconfig 的用法如下：

1. [root@localhost ~]# netconfig --help 注：帮助；
2. **--bootproto**=(dhcp|bootp|none) Boot protocol to use (
3. **--gateway**=STRING Network gateway (指定网关)
4. **--ip**=STRING IP address (指定 IP 地址)
5. **--nameserver**=STRING Nameserver (指定 DNS 客户端)
6. **--netmask**=STRING Netmask (指定网络掩码)
7. **--hostname**=STRING Hostname (指定主机名)
8. **--domain**=STRING Domain name (指定域名)
9. -d, **--device**=STRING Network device (指定网络设备)
10. --nodns No DNS lookups (没有 DNS 查询)
11. **--hwaddr**=STRING Ethernet hardware address (指定网卡的物理地址)
12. **--description**=STRING Description of the device (描述性文字)
13. Help options: (帮助选项)
14. -?, --help Show this help message
15. --usage Display brief usage message

实例一：设置网卡的 DHCP 模式自动获得 IP [root@localhost ~]# netconfig -d eth0
--bootproto=dhcp

实例一：手动设置网卡的 IP 等 [root@localhost ~]# netconfig -d eth0 --ip=192.168.1.33
--netmask=255.255.255.0 --gateway=192.168.1.1

4.3 Slackware 网卡配置文件和配置工具；

Slackware 有关网卡的配置文件是/etc/rc.d/rc.inet1.conf，这个文件包括以太网接口的网卡和无线网卡的配置。Slackware 还是比较纯净的，网络配置也较简单；在 Slackware 中也有 netconfig 配置工具，也是 text 模式运行的，人机交互界面，这个设置比较简单；

Slackware 用 netconfig 配置网卡完成后，其实质是修改了/etc/rc.d/rc.inet1.conf 文件。Slackware 是源法原味的 Linux 系统，他的配置文件比较标准，所以我推荐您在生产型的系统，不妨尝试一下 Slackware ；

配置好网卡后，我们还得运行下面的命令，才能激活网卡； [root@localhost ~]#
/etc/rc.d/rc.inet1 下面是一个例子，比如此机器有两个网卡 eth0 和 eth1，eth0 用 DHCP 获得 IP 地址，eth1 指定 IP 地址；

```
1. # Config information for eth0:
2. IPADDR[0]=""
3. NETMASK[0]=""
4. USE_DHCP[0]="yes" 注：在这里写上 yes，表示用 DHCP 获得 IP；
5. DHCP_HOSTNAME[0]="linuxsir01" 注：DNS 服务器主机名，也可以用 IP 来指定 DNS 服务器；
6. # Config information for eth1: 注：网卡 eth1 的配置；
7. IPADDR[1]="192.168.1.33" 注：指定 IP 地址；
8. NETMASK[1]="255.255.255.0" 注：指定掩码；
9. USE_DHCP[1]="no" 注：不用 DHCP 获得 IP；
10. DHCP_HOSTNAME[1]=""
11. # Config information for eth2:
12. IPADDR[2]=""
13. NETMASK[2]=""
14. USE_DHCP[2]=""
15. DHCP_HOSTNAME[2]=""
16. # Config information for eth3:
17. IPADDR[3]=""
18. NETMASK[3]=""
19. USE_DHCP[3]=""
20. DHCP_HOSTNAME[3]=""
21. # Default gateway IP address:
22. GATEWAY="192.168.1.1" 注：指定网关；
```

4.4 SuSE 或 OpenSuSE 网卡配置文件和配置工具；

正在更新之中；

5、关于拨号工具的介绍；

有的弟兄可能需要 ADSL 猫和普通猫的拨号工具；现在我们分别介绍一下；

5.1 ADSL pppoe 拨号工具 rp-pppoe；

如果您的 ADSL 不是路由的，如果是路由的，在路由路就能设置好自动拨号。只要把机器接上就能用了，这个咱们不说了，路由器大家都会用；但如果您的 ADSL 不支持路由，或您想用您当前所用的主机来做路由器；这就需要有一个拨号软件；

目前国内大多城市都用的是 pppoe 协议，所以我们有必要介绍 pppoe 拨号软件，在 Linux 中，这个软件的名字是 rp-pppoe；rp-pppoe 主页；

http://www.roaringpenguin.com/penguin/open_source_rp-pppoe.php

5.11 各大发行版自带的 rp-pppoe 的安装和使用；

rp-pppoe 目前在各大发行版本都是存在的, 比如 Redhat/Fedora、红旗、Slackware、Debian、SuSE 等系统, 都是采用这个拨号软件, 所以您大可不必为下载源码编译安装。只需要在各大发行版的安装盘中就可以找得到; 请用各大发行版自带的软件包管理工具来安装此软件包;

如果您用的是各大发行版提供的 rp-pppoe 软件包 比如 RPM 包的系统是用 `rpm -ivh rp-pppoe*.rpm` Slackware 系统是用 `installpkg rp-pppoe*.tgz` 在 Redhat/Fedora 中可以通过图形配置工具来完成, `redhat-config-network` 命令, 调用配置网络, 要通过 XDSL 来添加拨号, 比较简单;

所有发行版通用的方法是 `adsl-setup` 命令来配置 ADSL;

1. `[root@localhost ~]# adsl-setup` 注: 配置 pppoe 拨号, 请文档下面, 都差不多;
2. `[root@localhost ~]# adsl-start` 注: 启动拨号;
3. `[root@localhost ~]# adsl-stop` 注: 断开连接;

5.12 如果是源码包安装, 我们要自己来编译安装;

```
[root@localhost ~]# tar zxvf rp-pppoe-3.8.tar.gz
```

```
[root@localhost ~]# cd rp-pppoe-3.8
```

```
[root@localhost rp-pppoe-3.8]# ./go
```

```
Welcome to the Roaring Penguin PPPoE client setup. First, I will run  
  
some checks on your system to make sure the PPPoE client is installed  
properly...
```

```
Looks good! Now, please enter some information:
```

```
USER NAME
```

```
>>> Enter your PPPoE user name (default bxxnxbnx@sympatico.ca): 在这里添写你的拨号用户名; 就是服务商提供的;
```

```
>>> Enter the Ethernet interface connected to the DSL modem
```

```
For Solaris, this is likely to be something like /dev/hme0.
```

```
For Linux, it will be ethn, where 'n' is a number.
```

```
(default eth0): eth0 如果是以太网接口的 ADSL, 就要在这里写上接猫的那个网络接口号。此处是 eth0;
```

```
Do you want the link to come up on demand, or stay up continuously?
```

```
If you want it to come up on demand, enter the idle time in seconds
```

after which the link should be dropped. If you want the link to stay up permanently, enter 'no' (two letters, lower-case.)

NOTE: Demand-activated links do not interact well with dynamic IP addresses. You may have some problems with demand-activated links.

>>> Enter the demand value (default no): 注：默认回车

>>> Enter the DNS information here: 202.96.134.133 注：在这里写上 DNS 服务器地址；可以和提供商要，也可以用我写的这个；

Please enter the IP address of your ISP's secondary DNS server.

If you just press enter, I will assume there is only one DNS server.

>>> Enter the secondary DNS server address here: 202.96.128.143 这是第二个 DNS 服务器地址；

>>> Please enter your PPPoE password: 在这里输入用户的密码；

>>> Please re-enter your PPPoE password: 确认密码；

The firewall choices are:

0 - NONE: This script will not set any firewall rules. You are responsible for ensuring the security of your machine. You are STRONGLY recommended to use some kind of firewall rules.

1 - STANDALONE: Appropriate for a basic stand-alone web-surfing workstation

2 - MASQUERADE: Appropriate for a machine acting as an Internet gateway for a LAN

>>> Choose a type of firewall (0-2): 2 注：在这里写上 2，可以共享上网的；当然还得加一条防火墙规划；

Ethernet Interface: eth0

User name: dxxx

Activate-on-demand: No

Primary DNS: 202.96.134.133

Secondary DNS: 202.96.128.143

Firewalling: MASQUERADE

>>> Accept these settings and adjust configuration files (y/n)? y 注：是不是保存配置；

关于共享上网，请参考：《ADSL 共享上网的解决办法》

5.2 普通猫的拨号工具介绍；

普通猫分为串口和 PCI 的，请查看《关于网络设备概述》普通猫的拨号工具主要有 kppp 和 wvdial；在 Redhat/Fedora 中，用 system-config-network 或 redhat-config-network 也能设置 ppp 拨号；在 KDE 桌面环境下，大家一般都用 kppp 拨号工具，点鼠标就可以完成；

wvdial 工具是文本的，几乎在各大发行版都有。wvdial 的配置文件是 /etc/wvdial.conf。如果您的猫已经驱动好了，运行一下 wvdialconf 命令就生成了 /etc/wvdial.conf 了。当然您得查看一下它的内容：

1. [root@localhost ~]# wvdialconf
2. [root@localhost ~]# more /etc/wvdial.conf

关于 wvdial 工具的使用，请查看《普通 56K 猫拨号上网工具 wvdial 介绍》

6、无线网卡；

正在更新之中；由于我没有这样的网卡，是否有弟兄写一篇详尽一点的？在所有涉及无线网卡的文档中，这块都是空白。缺的就是这个。看来我是得弄一块无线网卡了。。。。。

7、DNS 客户端配置文件/etc/resolv.conf；

本来不应该把 DNS 客户端配置文件放在这里来说，但由于新手弟兄上网时，虽然能拨号，但不能以域名访问。究其原因是由于没有修改 /etc/resolv.conf 文件；

/etc/resolv.conf 里面存放的是各大通信公司 DNS 服务器列表；下面的三个地址可以用一用；当然您可以打电话问你的服务商；

1. nameserver 202.96.134.133
2. nameserver 202.96.128.143
3. nameserver 202.96.68.38

本文写了常用的以太网接口的配置，介绍了 Linux ifconfig、netconfig 等，我感觉最重要的还是配置文件，新手弟兄还是仔细看看配置文件吧。当您用工具配置完成后，不妨查看一下相应配置文件的变化。我认为这样的学习方式，能知其然，然后知所以然；

7.10、tcpdump

tcpdump 是一个用于截取网络分组，并输出分组内容的工具。tcpdump 凭借强大的功能和灵活的截取策略，使其成为类 UNIX 系统下用于网络分析和问题排查的首选工具。

tcpdump 提供了源代码，公开了接口，因此具备很强的可扩展性，对于网络维护和入侵者都是非常有用的工具。tcpdump 存在于基本的 Linux 系统中，由于它需要将网络界面设置为混杂模式，普通用户不能正常执行，但具备 root 权限的用户可以直接执行它来获取网络上的信息。因此系统中存在网络分析工具主要不是对本机安全的威胁，而是对网络上的其他计算机的安全存在威胁。

7.1.1 概述

顾名思义，tcpdump 可以将网络中传送的数据包的“头”完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供 and、or、not 等逻辑语句来帮助你去掉无用的信息。

引用

```
# tcpdump -vv
```

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
11:53:21.444591 IP (tos 0x10, ttl 64, id 19324, offset 0, flags [DF], proto 6, length: 92) asptest.localdomain.ssh > 192.168.228.244.1858: P 3962132600:3962132652(52) ack 2726525936 win 1266
asptest.localdomain.1077 > 192.168.228.153.domain: [bad udp cksum 166e!] 325+ PTR? 244.228.168.192.in-addr.arpa. (46)
11:53:21.446929 IP (tos 0x0, ttl 64, id 42911, offset 0, flags [DF], proto 17, length: 151) 192.168.228.153.domain > asptest.localdomain.1077: 325 NXDomain q: PTR? 244.228.168.192.in-addr.arpa. 0/1/0 ns: 168.192.in-addr.arpa. (123)
11:53:21.447408 IP (tos 0x10, ttl 64, id 19328, offset 0, flags [DF], proto 6, length: 172) asptest.localdomain.ssh > 192.168.228.244.1858: P 168:300(132) ack 1 win 1266
347 packets captured
1474 packets received by filter
745 packets dropped by kernel
不带参数的 tcpdump 会收集网络中所有的信息包头，数据量巨大，必须过滤。
```

7.1.2、命令介绍

命令格式为：tcpdump [-nn] [-i 接口] [-w 储存档名] [-c 次数] [-Ae]
[-qX] [-r 文件] [所欲捕获的数据内容]

参数：

- nn，直接以 IP 及 Port Number 显示，而非主机名与服务名称。
- i，后面接要「监听」的网络接口，例如 eth0，lo，ppp0 等等的接口。
- w，如果你要将监听所得的数据包数据储存下来，用这个参数就对了。后面接文件名。
- c，监听的数据包数，如果没有这个参数，tcpdump 会持续不断的监听，直到用户输入 [ctrl]-c 为止。
- A，数据包的内容以 ASCII 显示，通常用来提取 WWW 的网页数据包资料。
- e，使用资料连接层（OSI 第二层）的 MAC 数据包数据来显示。

-q, 仅列出较为简短的数据包信息, 每一行的内容比较精简。

-X, 可以列出十六进制 (hex) 以及 ASCII 的数据包内容, 对于监听数据包内容很有用。

-r, 从后面接的文件将数据包数据读出来。那个「文件」是已经存在的文件, 并且这个「文件」是由 -w 所制作出来的。所欲捕获的数据内容: 我们可以专门针对某些通信协议或者是 IP 来源进行数据包捕获。

那就可以简化输出的结果, 并取得最有用的信息。常见的表示方法有。

'host foo', 'host 127.0.0.1': 针对单台主机来进行数据包捕获。

'net 192.168': 针对某个网段来进行数据包的捕获。

'src host 127.0.0.1' 'dst net 192.168': 同时加上来源(src)或目标(dst)限制。

'tcp port 21': 还可以针对通信协议检测, 如 tcp、udp、arp、ether 等。

除了这三种类型的关键字之外, 其他重要的关键字如下: gateway, broadcast, less, greater, 还有三种逻辑运算, 取非运算是 'not' '!', 与运算是 'and', '&&'; 或运算是 'or', '||';

范例一: 以 IP 与 Port Number 捉下 eth0 这个网卡上的数据包, 持续 3 秒

```
[root@linux ~]# tcpdump -i eth0 -nn
```

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

```
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
```

```
01:33:40.41 IP 192.168.1.100.22 > 192.168.1.11.1190: P 116:232(116) ack 1 win 9648
```

```
01:33:40.41 IP 192.168.1.100.22 > 192.168.1.11.1190: P 232:364(132) ack 1 win 9648
```

<==按下 [ctrl]-c 之后结束

```
6680 packets captured <==提取下来的数据包数量
```

```
14250 packets received by filter <==由过滤所得的总数据包数量
```

```
7512 packets dropped by kernel <==被核心所丢弃的数据包
```

至于那个在范例一所产生的输出中, 我们可以大概区分为几个字段, 现以范例一当中那行特殊字体行来说明一下:

- 01:33:40.41: 这个是此数据包被捕获的时间, “时:分:秒” 的单位。
- IP: 通过的通信协议是 IP。
- 192.168.1.100.22>: 传送端是 192.168.1.100 这个 IP, 而传送的 Port Number 为 22, 那个大于(>)的符号指的是数据包的传输方向。
- 192.168.1.11.1190: 接收端的 IP 是 192.168.1.11, 且该主机开启 port 1190 来接收。
- P 116:232(116): 这个数据包带有 PUSH 的数据传输标志, 且传输的数据为整体数据的 116~232 Byte, 所以这个数据包带有 116 Bytes 的数据量。
- ack 1 win 9648: ACK 与 Window size 的相关资料。

最简单的说法, 就是该数据包是由 192.168.1.100 传到 192.168.1.11, 通过的 port 是由 22 到 1190, 且带有 116 Bytes 的数据量, 使用的是 PUSH 的标记, 而不是 SYN 之类的主动联机标志。

接下来，在一个网络状态很忙的主机上面，你想要取得某台主机对你联机的数据包数据时，使用 `tcpdump` 配合管线命令与正则表达式也可以，不过，毕竟不好捕获。我们可以通过 `tcpdump` 的表达式功能，就能够轻易地将所需要的数据独立的取出来。在上面的范例一当中，我们仅针对 `eth0` 做监听，所以整个 `eth0` 接口上面的数据都会被显示到屏幕上，但这样不好分析，可以简化吗？例如，只取出 `port 21` 的联机数据包，可以这样做：

```
[root@linux ~]# tcpdump -i eth0 -nn port 21
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
01:54:37.96 IP 192.168.1.11.1240 > 192.168.1.100.21: . ack 1 win 65535
01:54:37.96 IP 192.168.1.100.21 > 192.168.1.11.1240:P 1:21(20) ack 1 win 5840
01:54:38.12 IP 192.168.1.11.1240 > 192.168.1.100.21: . ack 21 win 65515
01:54:42.79 IP 192.168.1.11.1240 > 192.168.1.100.21:P 1:17(16) ack 21 win 65515
01:54:42.79 IP 192.168.1.100.21 > 192.168.1.11.1240: . ack 17 win 5840
01:54:42.79 IP 192.168.1.100.21 > 192.168.1.11.1240: P 21:55(34) ack 17 win 5840
```

看！这样就仅取出 `port 21` 的信息，如果仔细看的话，你会发现数据包的传递都是双向的，Client 端发出请求而 Server 端则予以响应，所以，当然是有去有回了。而我们也就可以经过这个数据包的流向来了解到数据包运动的过程了。例如：

- 我们先在一个终端机窗口输入“`tcpdump -i lo -nn`”的监听。
- 再另开一个终端机窗口来对本机（`127.0.0.1`）登录“`ssh localhost`”，那么输出的结果会是如何？

```
[root@linux ~]# tcpdump -i lo -nn
1 tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
2 listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes
3 11:02:54.253777 IP 127.0.0.1.32936 >
127.0.0.1.22: S 933696132:933696132(0)
    win 32767
4 11:02:54.253831 IP 127.0.0.1.22 > 127.0.0.1.32936:
S 920046702:920046702(0)
    ack 933696133 win 32767
5 11:02:54.253871 IP 127.0.0.1.32936 > 127.0.0.1.22: . ack 1 win 8192
6 11:02:54.272124 IP 127.0.0.1.22 > 127.0.0.1.32936:
P 1:23(22) ack 1 win 8192
7 11:02:54.272375 IP 127.0.0.1.32936 > 127.0.0.1.22: . ack 23 win 8192
```

代码显示的头两行是 `tcpdump` 的基本说明，然后：

- 第 3 行显示的是来自 Client 端带有 SYN 主动联机的数据包。
- 第 4 行显示的是来自 Server 端，除了响应 Client 端之外（ACK），还带有 SYN 主动联机的标志。
- 第 5 行则显示 Client 端响应 Server 确定联机建立（ACK）。
- 第 6 行以后则开始进入数据传输的步骤。

从第 3~5 行的流程来看，熟不熟悉啊？没错。那就是 3 次握手的基础流程，有趣吧。不过 tcpdump 之所以被称为黑客软件之一远不止上面介绍的功能。上面介绍的功能可以用来作为我们主机的数据包联机与传输的流程分析，这将有助于我们了解到数据包的运作，同时了解到主机的防火墙设置规则是否有需要修订的地方。

还有更神奇的用法。当我们使用 tcpdump 在 Router 上面监听明文的传输数据时，例如 FTP 传输协议，你觉得会发生什么问题呢？我们先在主机端执行“tcpdump -i lo port 21 -nn -X”，然后再以 FTP 登录本机，并输入账号与密码，结果你就可以发现如下的状况：

```
[root@linux ~]# tcpdump -i lo -nn -X 'port 21'
0x0000:  4500 0048 2a28 4000 4006 1286 7f00 0001  E..H*(@.@.....
0x0010:  7f00 0001 0015 80ab 8355 2149 835c d825  ....U!I.\.%
0x0020:  8018 2000 fe3c 0000 0101 080a 0e2e 0b67  ....<.....g
0x0030:  0e2e 0b61 3232 3020 2876 7346 5450 6420  ...a220.(vsFTPd.
0x0040:  322e 302e 3129 0d0a                      2.0.1)..
0x0000:  4510 0041 d34b 4000 4006 6959 7f00 0001  E..A.K@.@.iY....
0x0010:  7f00 0001 80ab 0015 835c d825 8355 215d  ....\.%U!]
0x0020:  8018 2000 fe35 0000 0101 080a 0e2e 1b37  ....5.....7
0x0030:  0e2e 0b67 5553 4552 2064 6d74 7361 690d  ...gUSER.dmtsai.
0x0040:  0a                                          .
0x0000:  4510 004a d34f 4000 4006 694c 7f00 0001  E..J.O@.@.iL....
0x0010:  7f00 0001 80ab 0015 835c d832 8355 217f  ....\..2.U!.
0x0020:  8018 2000 fe3e 0000 0101 080a 0e2e 3227  ....>.....2'
0x0030:  0e2e 1b38 5041 5353 206d 7970 6173 7377  ...8PASS.mypassw
0x0040:  6f72 6469 7379 6f75 0d0a                  ordisyou..
```

上面的输出结果已经被简化过了，你需要自行在你的输出结果中搜索相关的字符串才行。从上面输出结果的特殊字体中，我们可以发现该 FTP 软件使用的是 vsFTPd，并且用户输入 dmtsai 这个账号名称，且密码是 mypasswordisyu。如果使用的是明文方式来传输你的网络数据呢？

另外你得了解，为了让网络接口可以让 tcpdump 监听，所以执行 tcpdump 时网络接口会启动在“混杂模式 (promiscuous)”，所以你会在 /var/log/messages 里面看到很多的警告信息，通知你说你的网卡被设置成为混杂模式。别担心，那是正常的。至于更多的应用，请参考 man tcpdump 了。

例题：如何使用 tcpdump 监听来自 eth0 适配卡且通信协议为 port 22，目标来源为 192.168.1.100 的数据包资料？

答：tcpdump -i eth0 -nn port 22 and src host 192.168.1.100

例题：如何使用 tcpdump 抓取访问 eth0 适配卡且访问端口为 tcp 9080？

答：tcpdump -i eth0 dst 172.168.70.35 and tcp port 9080

例题：如何使用 tcpdump 抓取与主机 192.168.43.23 或着与主机 192.168.43.24 通信报文，并且显示在控制台上

答：tcpdump -X -s 1024 -i eth0 host \ (192.168.43.23 or 192.168.43.24\) and host 172.16.70.35

注：必须指定网卡

7.1.3、tcpdump 的表达式介绍

表达式是一个正则表达式，tcpdump 利用它作为过滤报文的条件，如果一个报文满足表达式的条件，则这个报文将会被捕获。如果没有给出任何条件，则网络上所有的信息包 将会被截获。

在表达式中一般如下几种类型的关键字：

引用

第一种是关于类型的关键字，主要包括 host, net, port, 例如 host 210.27.48.2, 指明 210.27.48.2 是一台主机, net 202.0.0.0 指明 202.0.0.0 是一个网络地址, port 23 指明端口号是 23。如果没有指定类型，缺省的类型是 host。

第二种是确定传输方向的关键字，主要包括 src, dst, dst or src, dst and src, 这些关键字指明了传输的方向。举例说明，src 210.27.48.2, 指明 ip 包中源地址是 210.27.48.2, dst net 202.0.0.0 指明目的网络地址是 202.0.0.0。如果没有指明 方向关键字，则缺省是 src or dst 关键字。

第三种是协议的关键字，主要包括 fddi, ip, arp, rarp, tcp, udp 等类型。Fddi 指明是在 FDDI (分布式光纤数据接口网络)上的特定的网络协议，实际上它是” ether” 的别名，fddi 和 ether 具有类似的源地址和目的地址，所以可以将 fddi 协议包当作 ether 的包进行处理和分析。其他的几个关键字就是指明了监听的包的协议内容。如果没有指定任何协议，则 tcpdump 将会 监听所有协议的信息包。

除了这三种类型的关键字之外，其他重要的关键字如下：gateway, broadcast, less, greater, 还有三种逻辑运算，取非运算是 ‘not ’ ’!’ ‘, 与运算是’ and’, ’ &&’ ;或运算是’ or’ ,’ | |’; 这些关键字可以组合起来构成强大的组合条件来满足人们的需要。

四、输出结果介绍

下面我们介绍几种典型的 tcpdump 命令的输出信息

(1) 数据链路层头信息

使用命令：

```
#tcpdump -e host ICE
```

ICE 是一台装有 linux 的主机。它的 MAC 地址是 0: 90: 27: 58: AF: 1A H219 是一台装有 Solaris 的 SUN 工作站。它的 MAC 地址是 8: 0: 20: 79: 5B: 46; 上一条命令的输出结果如下所示：

引用

```
21:50:12.847509 eth0 < 8:0:20:79:5b:46 0:90:27:58:af:1a ip 60: h219.33357 > ICE. telnet 0:0(0) ack 22535 win 8760 (DF)
```

21: 50: 12 是显示的时间， 847509 是 ID 号，eth0 <表示从网络接口 eth0 接收该分组， eth0 >表示从网络接口设备发送分组， 8:0:20:79:5b:46 是主机 H219 的 MAC 地址， 它表明是从源地址 H219 发来的分组。 0:90:27:58:af:1a 是主机 ICE 的 MAC 地址， 表示该分组的目的地址是 ICE。 ip 是表明该分组是 IP 分组， 60 是分组的长度， h219.33357 > ICE. telnet 表明该分组是从主机 H219 的 33357 端口发往主机 ICE 的 TELNET(23)端口。 ack 22535 表明对序列号是 22535 的包进行响应。 win 8760 表明发送窗口的大小是 8760。

(2) ARP 包的 tcpdump 输出信息

使用命令：

```
#tcpdump arp
```

得到的输出结果是：

引用

```
22:32:42.802509 eth0 > arp who-has route tell ICE (0:90:27:58:af:1a)
```

```
22:32:42.802902 eth0 < arp reply route is-at 0:90:27:12:10:66 (0:90:27:58:af:1a)
```

22:32:42 是时间戳， 802509 是 ID 号， eth0 > 表明从主机发出该分组， arp 表明是 ARP 请求包， who -has route tell ICE 表明是主机 ICE 请求主机 route 的 MAC 地址。 0:90:27:58:af:1a 是主机 ICE 的 MAC 地址。

(3) TCP 包的输出信息

用 tcpdump 捕获的 TCP 包的一般输出信息是：

引用

```
src > dst: flags data-seqno ack window urgent options
```

src > dst: 表明从源地址到目的地址， flags 是 TCP 报文中的标志信息， S 是 SYN 标志， F (FIN)， P (PUSH)， R (RST) “.” (没有标记)； data-seqno 是报文中的数据 的顺序号， ack 是下次期望的顺序号， window 是接收缓存的窗口大小， urgent 表明 报文中是否有紧急指针。 Options 是选项。

(4) UDP 包的输出信息

用 tcpdump 捕获的 UDP 包的一般输出信息是：

引用

```
route.port1 > ICE.port2: udp lenth
```

UDP 十分简单，上面的输出行表明从主机 route 的 port1 端口发出的一个 UDP 报文 到主机 ICE 的 port 2 端口，类型是 UDP， 包的长度是 lenth。

五、举例

(1) 想要截获所有 210.27.48.1 的主机收到的和发出的所有的分组：

```
#tcpdump host 210.27.48.1
```

(2) 想要截获主机 210.27.48.1 和主机 210.27.48.2 或 210.27.48.3 的通信，使用命令（注意：括号前的反斜杠是必须的）：

```
#tcpdump host 210.27.48.1 and \ (210.27.48.2 or 210.27.48.3 \)
```

(3) 如果想要获取主机 210.27.48.1 除了和主机 210.27.48.2 之外所有主机通信的 ip 包，使用命令：

```
#tcpdump ip host 210.27.48.1 and ! 210.27.48.2
```

(4) 如果想要获取主机 192.168.228.246 接收或发出的 ssh 包，并且不转换主机名使用如下命令：

```
#tcpdump -nn -n src host 192.168.228.246 and port 22 and tcp
```

(5) 获取主机 192.168.228.246 接收或发出的 ssh 包，并把 mac 地址也一同显示：

```
# tcpdump -e src host 192.168.228.246 and port 22 and tcp -n -nn
```

(6) 过滤的是源主机为 192.168.0.1 与目的网络为 192.168.0.0 的报头：

```
tcpdump src host 192.168.0.1 and dst net 192.168.0.0/24
```

(7) 过滤源主机物理地址为 XXX 的报头：

```
tcpdump ether src 00:50:04:BA:9B and dst.....
```

（为什么 ether src 后面没有 host 或者 net？物理地址当然不可能有网络喽）。

(8) 过滤源主机 192.168.0.1 和目的端口不是 telnet 的报头，并导入到 tes.t.txt 文件中：

Tcpdump src host 192.168.0.1 and dst port not telnet -l > test.txt
ip icmp arp rarp 和 tcp、udp、icmp 这些选项等都要放到第一个参数的位置，用来过滤数据报的类型。

十五、 配置文件

1、配置文件介绍：

每个 Linux 程序都是一个可执行文件，它含有操作码列表，CPU 将执行这些操作码来完成特定的操作。例如，ls 命令是由 /bin/ls 文件提供的，该文件含有机指令的列表，在屏幕上显示当前目录中文件的列表时需要使用这些机器指令。几乎每个程序的行为都可以通过修改其配置文件来按照您的偏好或需要去定制。

Linux 中有没有一个标准的配置文件格式？

一句话，没有。不熟悉 Linux 的用户（一定）会感到沮丧，因为每个配置文件看起来都象是一个要迎接的新挑战。在 Linux 中，每个程序员都可以自由选择他或她喜欢的配置文件格式。可以选择的格式很多，从 /etc/shells 文件（它包含被一个换行符分开的 shell 的列表），到 Apache 的复杂的 /etc/httpd.conf 文件。

什么是系统配置文件？

内核本身也可以看成是一个“程序”。为什么内核需要配置文件？内核需要了解系统中用户和组的列表，进而管理文件权限（即根据权限判定特定用户（UNIX_USERS）是否可以打开某个文件）。注意，这些文件不是明确地由程序读取的，而是由系统库所提供的一个函数读取，并被内核使用。例如，程序需要某个用户的（加密过的）密码时不应该打开 /etc/passwd 文件。相反，程序应该调用系统库的 getpw() 函数。这种函数也被称为系统调用。打开 /etc/passwd 文件和之后查找那个被请求的用户的密码都是由内核（通过系统库）决定的。

除非另行指定，Red Hat Linux 系统中大多数配置文件都在 /etc 目录中。配置文件可以大致分为下面几类：

2、配置文件分类：

访问文件

<i>/etc/host.conf</i>	告诉网络域名服务器如何查找主机名。（通常是 /etc/hosts，然后就是名称服务器；可通过 netconf 对其进行更改）
<i>/etc/hosts</i>	包含（本地网络中）已知主机的一个列表。如果系统的 IP 不是动态生成，就可以使用它。对于简单的主机名解析（点分表示法），在请求 DNS 或 NIS 网络名称服务器之前， /etc/hosts.conf 通常会告诉解析程序先查看这里。
<i>/etc/hosts.allow</i>	请参阅 hosts_access 的联机帮助页。至少由 tcpd 读取。
<i>/etc/hosts.deny</i>	请参阅 hosts_access 的联机帮助页。至少由 tcpd 读取。

引导和登录 / 注销

<i>/etc/issue & /etc/issue.net</i>	这些文件由 mingetty（和类似的程序）读取，用来向从终端（issue）或通过 telnet 会话(issue.net)连接的用户显示一个“welcome”字符串。它们包括几行声明 Red Hat 版本号、名称和内核 ID 的信息。它们由
--	---

	rc.local 使用。
<i>/etc/redhat-release</i>	包括一行声明 Red Hat 版本号和名称的信息。由 rc.local 使用。
<i>/etc/rc.d/rc</i>	通常在所有运行级别运行,级别作为参数传送。例如,要以图形(Graphics)模式(X-Server)引导机器,请在命令行运行下面的命令: <code>init 5</code> 。运行级别 5 表示以图形模式引导系统。
<i>/etc/rc.d/rc.local</i>	非正式的。可以从 rc、rc.sysinit 或 /etc/inittab 调用。
<i>/etc/rc.d/rc.sysinit</i>	通常是所有运行级别的第一个脚本。
<i>/etc/rc.d/rc/rcX.d</i>	从 rc 运行的脚本 (X 表示 1 到 5 之间的任意数字)。这些目录是特定“运行级别”的目录。当系统启动时,它会识别要启动的运行级别,然后调用该运行级别的特定目录中存在的所有启动脚本。例如,系统启动时通常会在引导消息之后显示“entering run-level 3”的消息;这意味着 /etc/rc.d/rc3.d/ 目录中的所有初始化脚本都将被调用。

文件系统

内核提供了一个接口,用来显示一些它的数据结构,这些数据结构对于决定诸如使用的中断、初始化的设备和内存统计信息之类的系统参数可能很有用。这个接口是作为一个独立但虚拟的文件系统提供的,称为 /proc 文件系统。很多系统实用程序都使用这个文件系统中存在的值来显示系统统计信息。例如, /proc/modules 文件列举系统中当前加载的模块。lsmod 命令读取此信息,然后将其以人们可以看懂的格式显示出来。下面表格中指定的 mtab 文件以同样的方式读取包含当前安装的文件系统的 /proc/mount 文件。

<i>/etc/mtab</i>	<p>这将随着 /proc/mount 文件的改变而不断改变。换句话说,文件系统被安装和卸载时,改变会立即反映到此文件中。</p> <ol style="list-style-type: none"> 1. 文件格式 /etc/mtab 的格式和/etc/fstab 是一样的. 但这个文件不能算是用户配置文件,他是由系统维护的. 和/etc/fstab 的区别在于, fstab 是系统启动时需挂载的文件系统列表,而 mtab 是系统当前已挂载的文件系统列表,它由系统维护,在用户执行了 mount 或者 umount 命令后自动更新. 用户不应该对此文件作任何修改. 2. 安全性 /etc/mtab 的默认权限仍然是 644 3. 相关命令 mount umount smbmount
<i>/etc/fstab</i>	<ol style="list-style-type: none"> 1. 文件格式 /etc/fstab 记载了系统启动时自动挂载的文件系统。一行为一条记录。每条记录有 6 个字段,字段间用空格或者 tab 键分开。这六个字段分别是: 设备名称,挂载点(除交换分区为 swap 外,都必须是一个存在的目录名),文件系统类型, mount 选项,是否需要 dump (1 表示需要, 0 表示不需要),在 reboot 期间 fsck 检查的顺序(激活文件系统设定为 1, 其余文件系统设定为 2, 若设定为 0 表示该文件系统不需要被检查)。 <p>在 linux 和 windows 共存时,也许想开机自动挂载 windows 分区,那么就可以在这个文件里加上相应的记录。</p>

某些时候对硬盘分区作了调整以后，这里也需要做一些相应的修改。否则会出现一些问题。

可用的 mount 选项：

async

对该文件系统的所有 I/O 操作都异步执行

ro

该文件系统是只读的

rw

该文件系统是可读可写的

atime

更新每次存取 inode 的存取时间

auto

可以使用 -a 选项 mount

defaults

使用预设的选项：rw, suid, dev, exec, auto, nouser, async

dev

解释在文件系统上的字符或区块设备

exec

允许执行二进制文件

noatime

不要在这个文件系统上更新存取时间

noauto

这个文件系统不能使用 -a 选项来 mount

nodev

不要解释在文件系统上的字符或区块设备

noexec

不允许在 mounted 文件系统上执行任何的二进制文件。这个选项对于具有包含非它自己的二进制结构的文件系统服务器而言非常有用

nosuid

不允许 setuid 和 setgid 位发生作用。（这似乎很安全，但是在安装 suidperl 后，同样不安全）。

nouser

限制一般非 root 用户 mount 文件系统

remount

尝试重新 mount 已经 mounted 的文件系统。这通常是用来改变文件系统的 mount 标志，特别是让只读的文件系统变成可擦写的

suid

允许 setuid 和 setgid 位发生作用

sync

文件系统的所有 I/O 同步执行

user

允许一般非 root 用户 mount 文件系统。这个选项会应用 noexec, nosuid, nodev 这三个选项（除非在命令行上有指定覆盖这些设定的选项）。

3. 安全性

	<p>/etc/fstab 的默认权限是 644,所有者和所有组均为 root</p> <p>2. 相关命令</p> <p>mount</p> <p>df</p> <p>列举计算机当前“可以安装”的文件系统。这非常重要,因为计算机引导时将运行 mount -a 命令,该命令负责安装 fstab 的倒数第二列中带有“1”标记的每一个文件系统。</p>
<i>/etc/mtools.conf</i>	DOS 类型的文件系统上所有操作(创建目录、复制、格式化等等)的配置。

系统管理

<i>/etc/group</i>	<p>1. 文件格式</p> <p>/etc/group 存储了系统中所有用户的基本信息. 它的格式和/etc/passwd 的格式基本类似,这里就说简单一点,</p> <p>/etc/group 也是由一条条的记录组成. 每条记录分 4 个字段. 分别是组名,组口令,组 ID 和该组包含用户列表. 其中组口令不再使用(现在只是保留为 x). 最后一个域是一个用逗号分隔的用户名列表,这个组的成员就是在这里列出的所有用户.</p> <p>2. 安全性</p> <p>/etc/group 的默认权限是 644,所有者和所有组均为 root. 注意经常检察.</p> <p>3. 相关命令</p> <p>groupadd</p> <p>groupdel</p> <p>groupmod</p> <p>groups</p> <p>包含有效的组名称和指定组中包括的用户。单一用户如果执行多个任务,可以存在于多个组中。例如,如果一个“用户”是“project 1”工程组的成员,同时也是管理员,那么在 group 文件中他的条目看起来就会是这样的: user:* : group-id : project1</p>
<i>/etc/nologin</i>	<p>这是一个普通的文本文件. 你可以在里面写上你喜欢的任何东西. /etc/nologin 的作用在于,如果它存在,那么系统将拒绝任何非 root 用户的登录请求,并对其它登录用户显示此文件的内容</p> <p>此文件常由系统在停机前自动生成. 有时系统管理员也会手工生成它,用以禁止其它用户登录,方便进行一些管理工作.</p>
<i>etc/passwd</i>	<p>1. 文件格式</p> <p>/etc/passwd 存储了系统中所有用户的基本信息. 可以说这是系统中最重要的一个配置文件. 对它作任何修改一定要小心谨慎. 同时要经常检察这个文件,包括它的内容和权限设置.</p> <p>使用 vi 编辑程序打开此文件,可以看到这个文件由许多行记录组成. 每一行记录对应着一个用户. 我们以第一行为例. 第一行一般是 root 用户的记录,尽管这不是必需的. 实际上用户记录出现的顺序并没有任何的意义.</p> <p>在我的系统中, /etc/passwd 的第一行看起来是这样的:</p> <p>root:x:0:0:root:/root:/bin/bash</p> <p>每一条记录都由 7 个字段组成,每个字段之间用冒号隔开. 第一个字段是用户名,示例中是 root. 第二个字段是用户口令,示例中是一个字符 x,但这并不表示</p>

	<p>root 的口令是单个字符 x, 而是说用户口令被加密了, 并且加密口令也没有放在本文件中, 而是放到了/etc/shadow(参考 /etc/shadow). 假如删除这个 x, 那么 root 的口令就清空了. 第三个字段是用户的用户 ID, 即 uid. 第四个字段是用户的组 ID, 即 gid. 这里要注意, 系统分辨两个用户是看他们的 uid 是否相同而不是看他们的用户名是否相同. 用户名不同但 uid 相同的两个用户实际上是同一个用户. 对组来说也有类似的规则. 所以这两个字段大家一定要注意. 第五个字段是用户全称, 没有什么实际用途, 相当于注释, 这里是 root. 第六个字段是用户的主目录 (home), 即登录系统后默认所处目录, 这里是/root. 最后一个字段是用户的登录 shell, 可以是系统拥有的任何一个 shell 的完整路径, 这里是 /bin/bash. 注意, 这个字段可以有一个特殊的值, 即/sbin/nologin. 如果把一个用户的登录 shell 设置为 /sbin/nologin 的话, 系统将禁止此用户的本地登录. 请参阅 “man passwd”. 它包含一些用户帐号信息, 包括密码(如果未被 shadow 程序加密过)。</p> <p>2. 安全性</p> <p>/etc/passwd 的默认权限为 644, 所有者和所有组均为 root. 切记, 在任何情况下都不要更改它.</p> <p>3. 相关命令</p> <p>passwd useradd userdel adduser usermod users</p>
<i>/etc/rpmrc</i>	<p>rpm 命令配置。所有的 rpm 命令行选项都可以在这个文件中一起设置, 这样, 当任何 rpm 命令在该系统中运行时, 所有的选项都会全局适用。</p>
<i>/etc/securetty</i>	<p>包含设备名称, 由 tty 行组成(每行一个名称, 不包括前面的 /dev/), root 用户在这里被允许登录。</p> <p>1. 文件格式</p> <p>这是一个设备文件的列表. 文件名取相对于/dev 的相对路径. 如, /dev/tty1 记为 tty1</p> <p>root 只有从这个列表中列出的设备上才可以登录系统.</p> <p>例如:</p> <p>代码:</p> <div><pre>\$cat /etc/securetty tty1 tty2 tty3</pre></div> <p>这里 root 被限定只能从/dev/tty1, /dev/tty2, /dev/tty3 这三个设备上登录系统</p> <p>如果/etc/securetty 不存在的话, 那么 root 将可以从任何设备登录系统.</p> <p>2. 安全性</p> <p>/etc/securetty 的默认权限是 600, 所有者和所有组都是 root</p>
<i>/etc/shadow</i>	<p>包含加密后的用户帐号密码信息, 还可以包括密码时效信息。包括的字段有:</p>

- 登录名
- 加密后的密码
- 从 1970 年 1 月 1 日到密码最后一次被更改的天数
- 距密码可以更改之前的天数
- 距密码必须更改之前的天数
- 密码到期前用户被警告的天数
- 密码到期后帐户被禁用的天数
- 从 1970 年 1 月 1 日到帐号被禁用的天数

1. 文件格式

/etc/shadow 文件保存的是用户名, 密码, 用户账号设置相关信息。

例:

```
root:$1$6UviCNvh$WTR0zPMek41K mzD0Z1DdV1:12264:3:4:5:6:12267:
```

第一段: root----- 用户注册名

第二段: \$1\$6UviCNvh\$WTR0zPMek41K mzD0Z1DdV1 ----加密口令

第三段: 12264-----上次更动密码的日期, 以 1970 年 1 月 1 日为 1, 1 天加 1

第四段: 3-----密码将被允许修改之前的天数 (0 表示“可在任何时间修改”)

第五段: 4-----系统将强制用户修改为新密码之前的天数 (1 表示“永远都不能修改”)

第六段: 5-----密码过期之前, 用户将被警告过期的天数 (-1 表示“没有警告”)

第七段: 6-----密码过期之后, 系统自动禁用帐户的天数 (-1 表示“永远不会禁用”)

第八段: 12267-----该帐户被禁用的天数 (-1 表示“该帐户被启用”). 以 1970 年 1 月 1 日为 1, 1 月 2 日为 2

第九段 ----- 保留供将来使用

注: 第 2 段中为*表示帐号不可登录, 如密码前为 !! 或只有 !! 表示帐号被锁

2. 安全性

	<p>/etc/shadow 的默认所有者和所有组均为 root.</p> <p>建议运行# <code>chattr +i /etc/shadow</code> 来保护文件使其不被意外地删除或重写</p> <p>3. 相关命令</p> <p><code>passwd</code></p> <p><code>useradd</code></p> <p><code>userdel</code></p> <p><code>usermod</code></p>
/etc/gshadow	<p>1. 文件格式</p> <p>/etc/gshadow 文件保存的是用户和组群设置的信息</p> <p>例:</p> <p><code>root:!!:root,wal</code></p> <p>第一段: 组名</p> <p>第四段: 该组包含用户列表</p> <p>2. 安全性</p> <p>/etc/gshadow 的默认所有者和所有组均为 root.</p> <p>建议运行# <code>chattr +i /etc/shadow</code> 来保护文件使其不被意外地删除或重写</p> <p>3. 相关命令</p> <p><code>groupadd</code></p> <p><code>groupdel</code></p> <p><code>groupmod</code></p> <p><code>groups</code></p>
/etc/sysctl.conf	<p>1. 文件格式</p> <p>/etc/sysctl.conf 是 sysctl 程序的配置文件. sysctl 可以在系统运行时更改内核参数. /etc/sysctl.conf 中的配置将在系统启动时执行.</p> <p>以 # 和 ; 开始的行是注释, 将和空白行一起被忽略.</p> <p>配置项的格式为:</p> <p><code>token = value</code></p> <p>token 是一个键名, value 是对应的键值. token 和 value 前后的空格将被忽略</p> <p>token 不能是随意的字符串. 他和 /proc/sys 下的文件有一一对应的关系:</p> <p>假设 foo 是 /proc/sys 下的一个文件. 删除 foo 的绝对路径前的 "/proc/sys" 这一部分, 然后把剩下部分中的 "/" 替换成 ".", 得到的字符串就是 foo 所对应的键名. 例如:</p> <p>/proc/sys/net/ipv4/ip_forward 对应的键名为 net.ipv4.ip_forward</p> <p>应用举例:</p> <p>Redhat Linux 9 默认是禁止 ip 转发的, 而我们在做 ip 伪装时需要起用 ip 转发. 通常的做法是在 iptables 的规则之前加上一句:</p> <p><code>echo 1>/proc/sys/net/ipv4/ip_forward</code></p> <p>实际上我们也可以在 /etc/sysctl.conf 中写上:</p>

	<p>net.ipv4.ip_forward = 1</p> <p>这样系统就默认起用 ip 转发了. 当然他不会立即生效. 因为/etc/sysctl.conf 是在系统起动时读入的. 想要立即生效的话, 请使用 sysctl 命令.</p> <p>2. 安全性</p> <p>/etc/sysctl.conf 的默认权限是 644, 所有者和所有组均为 root</p> <p>3. See also</p> <p>sysctl(8)</p> <p>sysctl.conf(5)</p> <p>proc(5)</p> <p>procinfo(8)</p>
<i>/etc/shells</i>	包含系统可用的可能的“shell”的列表。
<i>/etc/motd</i>	每日消息; 在管理员希望向 Linux 服务器的所有用户传达某个消息时使用。

联网

<i>/etc/gated.conf</i>	gated 的配置。只能被 gated 守护进程所使用。
<i>/etc/gated.version</i>	包含 gated 守护进程的版本号。
<i>/etc/gateway</i>	由 routed 守护进程可选地使用。
<i>/etc/networks</i>	列举从机器所连接的网络可以访问的网络名和网络地址。通过路由命令使用。允许使用网络名称。
<i>/etc/protocols</i>	列举当前可用的协议。请参阅 NAG（网络管理员指南，Network Administrators Guide）和联机帮助页。C 接口是 getprotoent。绝不能更改。
<i>/etc/resolv.conf</i>	<p>在程序请求“解析”一个 IP 地址时告诉内核应该查询哪个名称服务器。</p> <p>1. 文件格式</p> <p>/etc/resolv.conf 是系统的 DNS 解析器配置文件，最常见的用途是用来指定系统所使用的 DNS 服务器地址，您可以最多指定 MAXNS 个 DNS 服务器，MAXNS 是一个常量，在 /usr/include/resolv.h 中定义，一般为 3。每个 DNS 服务器地址应该以点分十进制格式写在单独的行上，前面加上关键字 nameserver。例如：</p> <pre>nameserver 173.26.100.99 nameserver 202.118.224.101</pre> <p>这里我们指定了两个 DNS 服务器，ip 地址分别为 173.26.100.99 和 202.118.224.101。当系统需要进行 DNS 解析时，优先使用列在前面的 DNS Server，如果解析失败则转而使用下一个 DNS Server。</p> <p>2. 安全性</p> <p>/etc/resolv.conf 的默认权限为 0644</p>
<i>/etc/host.conf</i>	1. 文件格式

	<p>/etc/host.conf 也是一个 DNS 解析器配置文件，但它最常见的用途是用来指定解析器使用的方法。一般来说，DNS 解析可以使用两种方法，一是查询 DNS 服务器，二是使用本地 hosts 主机表。/etc/host.conf 可以用来指定优先使用哪一种方法。可以使用 order 关键字来指定他们的优先级。order 后可跟一种或多种 DNS 查询方法，之间用逗号隔开，其优先级依次降低。可用的 DNS 查询方法有：hosts, bind, nis，分别表示使用本地 hosts 主机表，DNS 服务器，NIS 服务器来进行 DNS 查询。最常见的配置是： order bind, hosts</p> <p>2. 安全性</p> <p>/etc/host.conf 的默认权限为 0644</p>
<i>/etc/rpc</i>	包含 RPC 指令 / 规则，这些指令 / 规则可以在 NFS 调用、远程文件系统安装等中使用。
<i>/etc/exports</i>	要导出的文件系统（NFS）和对它的权限。
<i>/etc/services</i>	将网络服务名转换为端口号 / 协议。由 inetd、telnet、tcpdump 和一些其它程序读取。有一些 C 访问例程。
<i>/etc/inetd.conf</i>	inetd 的配置文件。请参阅 inetd 联机帮助页。包含每个网络服务的条目，inetd 必须为这些网络服务控制守护进程或其它服务。注意，服务将会运行，但在 /etc/services 中将它们注释掉了，这样即使这些服务在运行也将不可用。格式为：<service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
<i>/etc/sendmail.cf</i>	邮件程序 sendmail 的配置文件。比较隐晦，很难理解。
<i>/etc/sysconfig/network</i>	指出 NETWORKING=yes 或 no。至少由 rc.sysinit 读取。
<i>/etc/sysconfig/network-scripts/if*</i>	Red Hat 网络配置脚本。

系统命令

系统命令要独占地控制系统，并让一切正常工作。所有如 login（完成控制台用户身份验证阶段）或 bash（提供用户和计算机之间交互）之类的程序都是系统命令。因此，和它们有关的文件也特别重要。这一类别中有下列令用户和管理员感兴趣的文件。

<i>/etc/lilo.conf</i>	包含系统的缺省引导命令行参数，还有启动时使用的不同映象。您在 LILO 引导提示的时候按 Tab 键就可以看到这个列表。
<i>/etc/logrotate.conf</i>	维护 /var/log 目录中的日志文件。
<i>/etc/identd.conf</i>	identd 是一个服务器，它按照 RFC 1413 文档中指定的方式实现 TCP/IP 提议的标准 IDENT 用户身份识别协议。identd 的操作原理是查找特定 TCP/IP 连接并返回拥有此连接的进程的用户名。作为选择，它也可以返回其它信息，而不是用户名。请参阅 identd 联机帮助页。
<i>/etc/ld.so.conf</i>	“动态链接程序”（Dynamic Linker）的配置。
<i>/etc/inittab</i>	按年代来讲，这是 UNIX 中第一个配置文件。在一台 UNIX 机器打开之后启动的第一个程序是 init，它知道该启动什么，这是由于 inittab 的存在。

在运行级别改变时, init 读取 inittab, 然后控制主进程的启动。

1. 文件格式

init 进程将查看此文件来启动子进程, 完成系统引导. /etc/inittab 描述了一个进程是在系统引导时启动还是在系统引导完成后的某个情形下启动. 他也是由一行行的记录组成的. 而以 # 开头的行是注释, 将被忽略.

记录的格式是:

id:runlevels:action:process

id 域是一个由 1 到 4 个字符组成的字符串, 这个字符串必需是唯一的, 即不能有两条记录拥有相同的 id 域. id 域是一个标志域, 由它区分各条记录. 注意, 对于 gettys 或者其他的 login 进程来说, id 域必须是对应 tty 的 tty 后缀, 例如, 对于 tty1 来说, id 域应该是 1. 查看你的 /etc/inittab, 会发现类似下面这样的记录:

.....

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

.....

runlevels 域是一个运行级的列表, 可用的运行级有:

0 ---- 停机

1 ---- 单用户模式

2 ---- 不带 NFS 的多用户模式

3 ---- 完整的多用户模式

4 ---- 没有使用

5 ---- X11

6 ---- 重起系统

S ---- 单用户

s ---- 同 S

action 域是一个预定义的动作, 可用的 action 有:

respawn

进程终止后立刻重新开始(如 getty 进程)

wait

进程在进入指定的运行级后启动一次, 然后 init 将等待它的终止

once

进程在进入指定的运行级后启动一次

boot

进程在系统引导时启动, runlevels 域将被忽略

bootwait

进程在系统引导时启动, 然后 init 将等待它的终止, runlevels 域将被忽略

off

这个 action 不做任何事

ondemand

有一个特殊的运行级叫做 ondemand runlevel, 包括 a, b 和 c. 如果一个进程被标记了 ondemand runlevel, 那么当要求切换到这个 ondemand runlevel 时将会启动这个进程. 但实际上的 runlevel 不会改变

	<p><code>initdefault</code> 标记了 <code>initdefault</code> 这个 action 的记录项的 <code>runlevel</code> 域指定了系统引导完成后进入的运行级</p> <p><code>sysinit</code> 在系统引导时起动这个进程. 而且在所有的 <code>boot</code> 和 <code>bootwait</code> 项之前起动. <code>process</code> 域将被忽略</p> <p><code>powerwait</code> 在电力中断时起动这个进程. 通常会由一个与连接到计算机的 UPS 系统对话的进程通知 <code>init</code> 电力切断. <code>init</code> 在继续之前将等待这个进程结束</p> <p><code>powerfail</code> 同 <code>powerwait</code> 类似, 但是 <code>init</code> 不会等待这个进程结束</p> <p><code>powerokwait</code> 一旦 <code>init</code> 被通知电力已经恢复, 将起动这个进程</p> <p><code>powerfailnow</code> 当 <code>init</code> 被告知 UPS 的电力亦将耗尽时起动这个进程</p> <p><code>ctrlaltdel</code> 当 <code>init</code> 接到 <code>SIGINT</code> 信号时起动这个进程. 一般是按下了 <code>ctrl+alt+del</code> 这个组合键</p> <p><code>kbrequest</code> 当一个特殊的键盘组合键被按下时起动这个进程</p> <p><code>process</code> 域指定了将运行的进程, 可以有参数. 如果这个域以 <code>+</code> 开头, 表明 <code>init</code> 将为这个进程更新 <code>utmp/wtmp</code> 记录.</p> <p>范例:</p> <p><code>id:3:initdefault:</code> 系统引导完成后进入运行级 3</p> <p><code>si::sysinit:/etc/rc.d/rc.sysinit</code> 系统引导时运行 <code>/etc/rc.d/rc.sysinit</code></p> <p><code>10:0:wait:/etc/rc.d/rc 0</code> 系统进入运行级 0 时执行 <code>/etc/rc.d/rc 0</code>, 这里 0 是参数</p> <p><code>ca::ctrlaltdel:/sbin/shutdown -t3 -r now</code> 捕获到 <code>ctrl+alt+del</code> 时运行 <code>/sbin/shutdown -t3 -r now</code>. 如果想禁用 <code>ctrl+alt+del</code> 这个组合键, 直接删除或注释掉这行</p> <p><code>pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"</code> 电力中断时执行 <code>/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"</code></p> <p><code>1:2345:respawn:/sbin/mingetty tty1</code> 进入运行级 2, 3, 4 或 5 时执行 <code>respawn:/sbin/mingetty tty1</code>. 注意指定的 action 是 <code>respawn</code>, 这也就是为什么我们在终端下 <code>logout</code> 后会立刻又出现一个 <code>login</code> 提示符</p> <p><code>x:5:respawn:/etc/X11/prefdm -nodaemon</code> 进入运行级 5 时执行 <code>/etc/X11/prefdm -nodaemon</code>, 指定的 action 是 <code>respawn</code></p> <p>2. 安全性</p> <p><code>/etc/inittab</code> 的权限是 644, 所有者和所有组均为 <code>root</code></p>
--	--

	<p>3. 相关命令</p> <p>init</p> <p>telinit</p> <p>更多内容请</p> <p>man init</p> <p>man inittab</p>
<i>/etc/termcap</i>	一个数据库，包含所有可能的终端类型以及这些终端的性能。

守护进程

守护进程是一种运行在非交互模式下的程序。一般来说，守护进程任务是和联网区域有关的：它们等待连接，以便通过连接提供服务。Linux 可以使用从 Web 服务器到 ftp 服务器的很多守护进程。

<i>/etc/syslog.conf</i>	<p>syslogd 是一种守护进程，它负责记录（写到磁盘）从其它程序发送到系统的消息。这个服务尤其常被某些守护进程所使用，这些守护进程不会有另外的方法来发出可能有问题存在的信号或向用户发送消息。</p> <p>1. 文件格式</p> <p>/etc/syslog.conf 是 syslog 守护程序的配置文件. syslog 守护程序为记录来自运行于系统之上的程序的消息提供了一种成熟的客户机 - 服务器机制。syslog 接收来自守护程序或程序的消息，根据优先级和类型将该消息分类，然后根据由管理员可配置的规则将它写入日志。结果是一个健壮而统一的管理日志的方法。</p> <p>这个文件由一条条的规则组成. 每条规则应该写在一行内. 但是如果某行以反斜线 \ 结尾的话, 他的下个物理行将被认为与此行同属于一行. 空白行和以 # 开始的行将被忽略.</p> <p>每条规则都是下面这种形式:</p> <p>facility.priority[;facility.priority] action</p> <p>facility 和 priority 之间用一个英文句点分隔. 他们的整体称为 selector. 每条规则可以有多个 selector, selector 之间用分号隔开. 而 selector 和 action 之间则用空格或者 tab 隔开.</p> <p>facility 指定 syslog 功能，主要包括以下这些:</p> <p>auth 由 pam_pwdb 报告的认证活动。</p> <p>authpriv 包括特权信息如用户名在内的认证活动</p> <p>cron 与 cron 和 at 有关的信息。</p> <p>daemon 与 inetd 守护进程有关的信息。</p> <p>kern 内核信息，首先通过 klogd 传递。</p> <p>lpr 与打印服务有关的信息。</p> <p>mail 与电子邮件有关的信息</p> <p>mark syslog 内部功能用于生成时间戳</p> <p>news 来自新闻服务器的信息</p> <p>syslog 由 syslog 生成的信息</p> <p>user 由用户程序生成的信息</p> <p>uucp 由 uucp 生成的信息</p> <p>local0----local7 与自定义程序使用，例如使用 local5 做为 ssh 功能</p>
-------------------------	--

* 通配符代表除了 mark 以外的所有功能

priority 指定消息的优先级. 与每个功能对应的优先级是按一定顺序排列的, emerg 是最高级, 其次是 alert, 依次类推. 缺省时, 在 /etc/syslog.conf 记录中指定的级别为该级别和更高级别. 如果希望使用确定的级别可以使用两个运算符号! (不等)和=.

user.=info

表示告知 syslog 接受所有在 info 级别上的 user 功能信息.

可用的 syslog 优先级如下:

emerg 或 panic 该系统不可用

alert 需要立即被修改的条件

crit 阻止某些工具或子系统功能实现的错误条件

err 阻止工具或某些子系统部分功能实现的错误条件

warning 预警信息

notice 具有重要性的普通条件

info 提供信息的消息

debug 不包含函数条件或问题的其他信息

none 没有重要级, 通常用于排错

* 所有级别, 除了 none

action 字段所表示的活动具有许多灵活性, 特别是, 可以使用名称管道的作用是可以使 syslogd 生成后处理信息.

syslog 主要支持以下 action

file

指定文件的绝对路径, 如: /var/log/messages . log 信息将写到此文件中

terminal 或 printer

完全的串行或并行设备标志符, 如/dev/console . log 信息将送到此设备

@host

远程的日志服务器. log 信息将送到此日志服务器

username

发送信息给指定用户

named pipe

指定使用 mkfifo 命令来创建的 FIFO 文件的绝对路径.

如果对此文件作了改动, 想要使改动生效, 您需要向 syslog 守护程序通知所做的更改. 向它发送 SIGHUP 是个正确的办法, 您可以用 killall 命令轻松地做到这一点:

```
# killall -HUP syslogd
```

2. 安全性

您应该清楚如果 syslogd 写的日志文件还不存在的话, 程序将创建它们. 无论您当前的 umask 如何设置, 该文件将被创建为可被所有用户读取. 如果您关心安全性, 那么您应该用 chmod 命令将该文件设置为仅 root 用户可读写. 此外, 可以用适当的许可权配置 logrotate 程序 (在下面描述) 以创建新的日志文件. syslog 守护程序始终会保留现有日志文件的当前属性, 因此一旦创建了文件, 您就不需要担心它.

3. 相关命令

logrotate

	klogd syslogd dmesg
<i>/etc/httpd.conf</i>	Web 服务器 Apache 的配置文件。这个文件一般不在 /etc 中。它可能在 /usr/local/httpd/conf/ 或 /etc/httpd/conf/ 中，但是要确定它的位置，您还需要检查特定的 Apache 安装信息。
<i>/etc/conf.modules</i> or <i>/etc/modules.conf</i>	kernel 的配置文件。有意思的是，kernel 并不是“作为守护进程的”内核。它其实是一种在需要时负责“快速”加载附加内核模块的守护进程。

用户程序

在 Linux（和一般的 UNIX）中，有无数的“用户”程序。最常见的一种用户程序配置文件是 /etc/lynx.cfg。这是著名的文本浏览器 lynx 的配置文件。通过这个文件，您可以定义代理服务器、要使用的字符集等等。下面的代码样本展示了 lynx.cfg 文件的一部分，修改这部分代码可以改变 Linux 系统的代理服务器设置。缺省情况下，这些设置适用于在各自的 shell 中运行 lynx 的所有用户，除非某个用户通过指定 --cfg = "mylynx.cfg" 重设了缺省的配置文件。

/etc/lynx.cfg 中的代理服务器设置

```
.h1 proxy
.h2 HTTP_PROXY
.h2 HTTPS_PROXY
.h2 FTP_PROXY
.h2 GOPHER_PROXY
.h2 NEWS_PROXY
.h2 NNTP_PROXY
# Lynx version 2.2 and beyond supports the use of proxy servers that can act as
# firewall gateways and caching servers. They are preferable to the older
# gateway servers. Each protocol used by Lynx can be mapped separately using
# PROTOCOL_proxy environment variables (see Lynx Users Guide). If you have
# not set them externally, you can set them at run time via this configuration file.
# They will not override external settings. The no_proxy variable can be used
# to inhibit proxying to selected regions of the Web (see below). Note that on
# VMS these proxy variables are set as process logicals rather than symbols, to
# preserve lowercasing, and will outlive the Lynx image.
#
.ex 15
http_proxy:http://proxy3.in.ibm.com:80/
ftp_proxy:http://proxy3.in.ibm.com:80/
#http_proxy:http://penguin.in.ibm.com:8080
#ftp_proxy:http://penguin.in.ibm.com:8080/
.h2 NO_PROXY
# The no_proxy variable can be a comma-separated list of strings defining
# no-proxy zones in the DNS domain name space. If a tail substring of the
# domain-path for a host matches one of these strings, transactions with that
```

```
# node will not be proxied.  
.ex  
no_proxy:demiurge.in.ibm.com, demiurge
```

更改配置文件

在更改配置文件时，如果程序不是由系统管理员或内核控制的，就要确保重新启动过使用该配置的程序。普通用户通常没有启动或停止系统程序和 / 或守护进程的权限。

内核

更改内核中的配置文件会立即影响到系统。例如，更改 `passwd` 文件以增加用户将立即使用户变为可用。而且任何 Linux 系统的 `/proc/sys` 目录中都有一些内核可调参数。只有超级用户可以得到对所有这些文件的写访问权力；其它用户只有只读访问权力。此目录中文件的分类的方式和 Linux 内核源代码的分类方式一样。此目录中的每个文件都代表一个内核数据结构，这些数据结构可以被动态地修改，从而改变系统性能。

注意：在更改其中任何文件的任何值之前，您应该确保自己全面了解该文件，以避免对系统造成不可修复的损害。

`/proc/sys/kernel/` 目录中的文件

文件名	描述
<code>threads-max</code>	内核可运行的最大任务数。
<code>ctrl-alt-del</code>	如果值为 1, 那么顺序按下这几个键将“彻底地”重新引导系统。
<code>sysrq</code>	如果值为 1, <code>Alt-SysRq</code> 则为激活状态。
<code>osrelease</code>	显示操作系统的发行版版本号
<code>ostype</code>	显示操作系统的类型。
<code>hostname</code>	系统的主机名。
<code>domainname</code>	网络域，系统是该网络域的一部分。
<code>modprobe</code>	指定 <code>modprobe</code> 是否应该在启动时自动运行并加载必需的模块。

守护进程和系统程序

守护进程是永远运行在后台的程序，它默默地执行自己的任务。常见的守护进程有 `in.ftpd` (ftp 服务器守护进程)、`in.telnetd` (telnet 服务器守护进程) 和 `syslogd` (系统日志记录守护进程)。有些守护进程在运行时会严密监视配置文件，在配置文件改变时就会自动重新加载它。但是大多数守护进程并不会自动重新加载配置文件。我们需要以某种方式“告诉”这些守护进程配置文件已经被发生了改变并应该重新加载。可以通过使用服务命令重新启动服务来达到这个目的（在 Red Hat Linux 系统上）。

例如，如果我们更改了网络配置，就需要发出：

```
service network restart
```

注意: 这些服务最常见的是 `/etc/rc.d/init.d/*` 目录中存在的脚本，在系统被引导时由 `init` 启动。

所以，您也可以执行如下操作来重新启动服务：

```
/etc/rc.d/init.d/<script-for-the-service> start | stop | status
```

`start`、`stop` 和 `status` 是这些脚本接受的输入值，用来执行操作。

用户程序

用户或系统程序在每次启动时都会读取其配置文件。尽管如此，请记住，有些系统程序在计算机打开时情况不一样，它们的行为依赖于在 `/etc/` 中的配置文件中读到的内容。所以，用户程序第一次启动时将从 `/etc/` 目录中存在的文件读取缺省配置。然后，用户可以通过使用 `rc` 和 `.`（点）文件来定制程序，正如下面一节所示。

用户配置文件：`.`（点）文件和 `rc` 文件

我们已经看到怎样容易地配置程序。但是如果有的人不喜欢在 `/etc/` 中配置程序的方式该怎么办呢？

“普通”用户不能简单地进入 `/etc` 然后更改配置文件；从文件系统的角度来看，配置文件的所有者是 `root` 用户！这就是大多数用户程序都定义两个配置文件的原因：第一个是“系统”级别的，位于 `/etc/`；另一个属于用户“专用”，可以在他或她的主目录中找到。

例如，我在我的系统中安装了非常有用的 `wget` 实用程序。`/etc/` 中有一个 `/etc/wgetrc` 文件。在我的主目录中，有一个名为 `.wgetrc` 的文件，它描述了我定制的配置（只有在我，也就是用户运行 `wget` 命令时，才会加载这个配置文件）。其它用户在他们自己的主目录（`/home/other`）中也可以有 `.wgetrc` 文件；当然，只有这些用户运行 `wget` 命令时，才会读取这个文件。换句话说，`/etc/wgetrc` 文件为 `wget` 提供了“缺省”值，而 `/home/xxx/.wgetrc` 文件列举了某个用户的“定制项”。重要的是这只是“一般规则”，并非所有情况都如此。例如，一个象 `pine` 一样的程序，在 `/etc/` 中并没有任何文件，它只在用户主目录中有一个定制配置文件，名为 `.pinerc`。其它程序可能只有 `/etc/` 中的缺省配置文件，而且可能不允许用户“定制”这些配置文件（`/etc` 目录中只有少数 `config.` 文件是这种情况）。

通常使用的 `rc` 和 `.`（点）文件

文件名	描述
<code>~/.bash_login</code>	请参考“man bash”。如果 <code>~/.bash_profile</code> 不存在， <code>bash</code> 则将 <code>~/.bash_login</code> 作为 <code>~/.bash_profile</code> 处理。
<code>~/.bash_logout</code>	请参考“man bash”。在退出时由 <code>bash</code> 登录 <code>shell</code> 引用。
<code>~/.bash_profile</code>	由 <code>bash</code> 登录 <code>shell</code> 引用 <code>/etc/profile</code> 之后引用。
<code>~/.bash_history</code>	先前执行的命令的列表。
<code>~/.bashrc</code>	请参考“man bash”。由 <code>bash</code> 非登录交互式 <code>shell</code> 引用（没有其它文件）。除非设置了 <code>BASH_ENV</code> 或 <code>ENV</code> ，非交互式 <code>shell</code> 不引用任何文件。
<code>~/.emacs</code>	启动时由 <code>emacs</code> 读取。
<code>~/.forward</code>	如果这里包含一个电子邮件地址，那么所有发往 <code>~</code> 的所有者的邮件都会被转发到这个电子邮件地址。
<code>~/.fvwmrc</code> <code>~/.fvwm2rc</code>	<code>fvwm</code> 和 <code>fvwm2</code> （基本的 <code>X Window</code> 管理器）的配置文件。
<code>~/.hushlogin</code>	请参考“man login”。引起“无提示”登录（没有邮件通知、上次登录信息或者 <code>MOD</code> 信息）。

~/.mail.rc	邮件程序的用户初始化文件。
~/.ncftp/	ncftp 程序的目录；包含书签、日志、宏、首选项和跟踪信息。请参阅 man ncftp。ncftp 的目的是为因特网标准文件传输协议（Internet standard File Transfer Protocol）提供一个强大而灵活的接口。它旨在替换系统所使用的标准的 ftp 程序。
~/.profile	请参考“man bash”。如果 ~/.bash_profile 和 ~/.bash_login 文件不存在，bash 则将 ~/.profile 作为 ~/.bash_profile 处理，并被其它继承 Bourn 的 shell 使用。
~/.pinerc	Pine 配置
~/.muttrc	Mutt 配置
~/.exrc	这个文件可以控制 vi 的配置。 示例：set ai sm ruler 在此文件中写入上面一行会让 vi 设置自动缩进、匹配括号、显示行号和行-列这几个选项。
~/.vimrc	缺省的“Vim”配置文件。和 .exrc 一样。
~/.gtkrc	GNOME 工具包（GNOME Toolkit）。
~/.kderc	KDE 配置。
~/.netrc	ftp 缺省登录名和密码。
~/.rhosts	由 r- 工具（如 rsh、rlogin 等等）使用。因为冒充主机很容易，所以安全性非常低。 1. 必须由用户（~/ 的所有者）或超级用户拥有。 2. 列出一些主机，用户可以从这些主机访问该帐号。 3. 如果是符号链接则被忽略。
~/.rpmrc	请参阅“man rpm”。如果 /etc/rpmrc 不存在则由 rpm 读取。
~/.signature	消息文本，将自动附加在从此帐号发出的邮件末尾。
~/.twmrc	twm（ T he W indow M anager）的配置文件。

<code>~/.xinitrc</code>	启动时由 X 读取（而不是由 xinit 脚本读取）。通常会启动一些程序。 示例： <code>exec /usr/sbin/startkde</code> 如果该文件中存在上面这行内容，那么在从这个帐号发出 <code>startx</code> 命令时，这一行就会启动“KDE 视窗管理器”（KDE Window Manager）。
<code>~/.xmodmaprc</code>	此文件被传送到 <code>xmodmap</code> 程序，而且可以被命名为任何文件（例如 <code>~/.Xmodmap</code> 和 <code>~/.keymap.km</code> ）。
<code>~/.xserverrc</code>	如果 <code>xinit</code> 可以找到要执行的 X， <code>xinit</code> 就会将该文件作为 X 服务器运行。
<code>~/News/Sent-Message-IDs</code>	gnus 的缺省邮件历史文件。
<code>~/.Xauthority</code>	由 <code>xdm</code> 程序读和写，以处理权限。请参阅 X、 <code>xdm</code> 和 <code>xauth</code> 联机帮助页。
<code>~/.Xdefaults</code> , <code>~/.Xdefaults-hostname</code>	在主机 <code>hostname</code> 的启动过程中由 X 应用程序读取。如果找不到 <code>-hostname</code> 文件，则查找 <code>.Xdefaults</code> 文件。
<code>~/.Xmodmap</code>	指向 <code>.xmodmaprc</code> ；Red Hat 有使用这个名称的 <code>.xinitrc</code> 文件。
<code>~/.Xresources</code>	通常是传送到 <code>xrdb</code> 以加载 X 资源数据库的文件的名称，旨在避免应用程序需要读取一个很长的 <code>.Xdefaults</code> 文件这样的情况。（有些情况曾经使用了 <code>~/.Xres</code> 。）
<code>~/mbox</code>	用户的旧邮件。

3、重要的配置文件列表：

启动引导程序配置文件

LILO `/etc/lilo.conf`

GRUB `/boot/grub/menu.lst`

系统启动文件核脚本

主启动控制文件 `/etc/inittab`

SysV 启动脚本的位置 `/etc/init.d`、`/etc/rc.d/init.d` 或 `/etc/rc.d`

SysV 启动脚本链接的位置 `/etc/init.d/rc?.d`、`/etc/rc.d/rc?.d` 或 `/etc/rc?.d`

本地启动脚本 `/etc/rc.d/rc.local`、`/etc/init.d/boot.local` 或 `/etc/rc.boot` 里的文件

网络配置文件

建立网络接口的脚本 `/sbin/ifup`

保存网络配置数据文件的目录 `/etc/network`、`/etc/sysconfig/network` 和 `/etc/sysconfig/network-scripts`

保存解析 DNS 服务的文件 /etc/resolv.conf
DHCP 客户端的配置文件 /etc/dhclient.conf
超级服务程序配置文件和目录
inetd 配置文件 /etc/inetd.conf
TCP Wrappers 配置文件 /etc/hosts.allow 和/etc/hosts.deny
xinetd 配置文件 /etc/xinetd.conf 和/etc/xinetd.d 目录里的文件
硬件配置
内核模块配置文件 /etc/modules.conf
硬件访问文件
Linux 设备文件 /dev 目录里
保存硬件和驱动程序数据的文件 /proc 目录里
扫描仪配置文件
SANE 主配置 /etc/sane.d/dll.conf
特定扫描仪的配置文件 /etc/sane.d 目录里以扫描仪型号命名的文件
打印机配置文件
BSD LPD 核 LPRng 的本地打印机主配置文件 /etc/printcap
CUPS 本地打印机主配置和远程访问受权文件 /etc/cups/cupsd.conf
BSD LPD 远程访问受权文件 /etc/hosts.lpd
LPRng 远程访问受权文件 /etc/lpd.perms
文件系统
文件系统表 /etc/fstab
软驱装配点 /floppy、/mnt/floppy 或/media/floppy
光驱装配点 /cdrom、/mnt/cdrom 或/media/cdrom
shell 配置文件
bash 系统非登录配置文件 /etc/bashrc、/etc/bash.bashrc 或/etc/bash.bashrc.local
bash 系统登录文件 /etc/profile 和/etc/profile.d 里的文件
bash 用户非登录配置文件 ~/.bashrc
bash 用户登录配置文件 ~/.profile
XFree86 配置文件核目录
XFree86 主配置文件 /etc/XF86config、/etc/X11/XF86Config 或/etc/X11/XF86Config-4
字体服务程序配置文件 /etc/X11/fs/config
Xft 1.x 配置文件 /etcX11/XftConfig
Xft 2.0 配置文件 /etc/fonts/fonts.conf
字体目录 /usr/X11R6/lib/X11/fonts 和/usr/share/fonts
Web 服务程序配置文件
Apache 主配置文件 /etc/apache、/etc/httpd 或/httpd/conf 里的 httpd.conf 或 httpd2.conf 文件
MIME 类型文件 与 Apache 主配置文件在同一目录里的 mime.types 或 apache-mime.types
文件服务程序配置文件
ProFTPD 配置文件 /etc/proftpd.conf
vsftpd 配置文件 /etc/vsftpd.conf
NFS 服务程序的输出定义文件 /etc/exports
NFS 客户端装配的 NFS 输出 /etc/fstab
Samba 配置文件 /etc/samba/smb.conf

Samba 用户配置文件 /etc/samba/smbpasswd
邮件服务程序配置文件
sendmail 主配置文件 /etc/mail/sendmail.cf
sendmail 源配置文件 /etc/mail/sendmail.mc 或/usr/share/sendmail/cf/cf/linux.smtp.mc 或其他文件
Postfix 主配置文件 /etc/postfix/main.cf
Exim 主配置文件 /etc/exim/exim.cf
Procmail 配置文件 /etc/procmailrc 或 ~/.procmailrc
Fetchmail 配置文件 ~/.fetchmailrc
远程登录配置文件
SSH 服务程序配置文件 /etc/ssh/sshd_config
SSH 客户端配置文件 /etc/ssh/ssh_config
XDM 配置文件 /etc/X11/xdm 目录下
GDM 配置文件 /etc/X11/gdm 目录下
VNC 服务程序配置文件 /usr/X11R6/bin/vncserver 启动脚本和 ~/.vnc 目录里的文件
其他服务程序配置文件
DHCP 服务程序配置文件 /etc/dhcpd.conf
BIND 服务程序配置文件 /etc/named.conf 和/var/named/
NTP 服务程序配置文件 /etc/ntp.conf

十六、计划任务

在很多时候为了自动化管理系统，我们都会用到计划任务，比如关机，管理，备份之类的操作，我们都可以使用计划任务来完成，这样可以使管理员的工作量大大降低，而且可靠度更好。
linux 系统支持一些能够自动执行任务的服务，我们称为计划任务。

LINUX 有如下三种计划任务：

at：指定一个时间执行一个任务（适用一个或多个任务，执行一次后就不用）

cron：根据一个时间表自动执行任务（使用一个或多个任务，周期性执行）

系统级别的计划任务及其扩展 anacron：在一个指定时间间隔错过后自动执行任务

1、at：安排一个任务在未来执行，需要一个atd的系统后台进程

检查 atd 进程是否启动

```
[root@centos61 桌面]# service atd status
```

```
atd (pid 2274) 正在运行...
```

```
[root@centos61 桌面]# chkconfig |grep atd
```

```
atd          0:关闭  1:关闭  2:关闭  3:启用  4:启用  5:启用  6:关闭
```

如果未启动，可以使用如下命令：

```
[root@centos61 桌面]# service atd start
```

```
正在启动 atd:
```

```
[确定]
```

```
[root@centos61 桌面]# chkconfig atd on
```

常用指令： at:安排延时任务

具体使用方法：

例 1：

```
#at now+2 minutes 回车
```

```
>输入要执行的命令
```


>ctrl+d 结束输入

```
[root@test ~]# at now+2 minutes
```

```
at> wall Aixi
```

```
at> <EOT>
```

```
job 2 at 2010-06-18 16:36
```

<EOT>是 ctrl+d 中断输入，这个命令意思是发送一个广播内容是 Hello Aixi. 具体时间可以改, 单位可以改, 可以用 hours, months, years, weeks 等.

例 2 我们还可以跟具体时间

```
[root@test ~]# at 16:39 dec 10
```

```
at> Hello Aixi
```

```
at> <EOT> ctrl+d 结束输入
```

```
job 3 at 2010-12-10 16:39
```

意思是在今年的 12 月 10 日 16:39 运行这个命令. 如果不加月和日, 默认就是今天.

Atq: 查询当前的等待任务

用 atq 来查询, 已经运行的任务, 就消失了。这就是 at 计划任务的重点, 只运行一次

atrm: 删除等待任务

启动计划任务后, 如果不想启动设定好的计划任务可以使用 atrm 命令删除。

格式: atrm 任务号

命令后面跟计划任务编号, 如果不跟, 就会删除这个用户所有的计划任务。

例 3

```
atrm 10 //删除计划任务 10
```

```
atq //查看计划任务是否删除
```

at 将要运行的命令以文本形式写入/var/spool/at/目录内, 等待 atd 服务的取用和执行。

还可以进入到/var/spool/at 目录里把计划任务删除, 计划任务的文件都保存在该目录里, 可以用 rm -f 文件名来删除(以文件的形式删除计划任务, 因为计划任务是以文件形式保存在该目录中)

例 4:

```
#cd /var/spool/at //进入到/var/spool/at 目录中
```

```
ls //显示目录中所有文件
```

```
rm -f a0000b0138b19c //删除计划任务
```

在通常情况下, 超级用户都可以使用这个命令。对于其他用户来说, 能否可以使用就取决于两个文件:/etc/at.allow 和/etc/at.deny。

at 命令是可以基于用户来控制的, 我们可以明确指定哪些用户可以使用 at 计划任务, 哪些用户不可以使用 at 计划任务。

at 的控制文件

/etc/at.allow

/etc/at.deny

系统默认是有 at.deny 文件, 如果某个用户名在这个文件里, 他就不能使用 at 计划任务。如果有 at.allow 文件, allow 文件先行, 检查了 allow 明确允许, 就不会检查 deny。

如果你要让哪个用户不能使用计划任务, 就直接把他的用户名写进去就可以了, 一排只能写一个。

2、cron服务(参考网址: <http://www.linuxsir.org/main/?q=node/209>)

相对与 at, cron 的优点就是能够周期性的执行某个命令, at 却只能执行一次, cron 的后台进程名字是 crond, cron 也是 system V 的服务, 所以我们可以 service crond start|stop 来启动和关闭此服务, 也可以使用 chkconfig 或者 ntsysv 来选择 cron 服务的默认开启, 这些命令在以前我们都讲过的命令:

```
#crontab -e          编辑当前用户的 cron 表
#crontab -l          查看当前用户的 cron 表
#crontab -r          删除当前用户的 cron 进程
#crontab -u 用户名 以某用户的身份来控制 cron 表
```

还有个重要的知识点, 就是当用户的计划任务建立后是存放在 var/spool/cron 这个目录

当使用 crontab -e 编辑当前用户的 cron 表后, 会出现一个 vi 文件, cron 的格式是这样的。分成两列, 左边是时间, 右边是运行的命令。时间是由 5 个部分组成。

例:

```
* * * * *          wall hello everyone
```

5 个星号分别代表: minute hour day-of-month month-of-year day-of-week , 而 wall hello everyone 这是命令内容。上面的意思是每分每小时每天每月每周广播 hello everyone。具体时间大家可以自己定义。如果要每两分钟发送就用*/2 代替第一个*。也可以是用具体时间来表示。

我们使用 crontab -e 编辑当前用户的 cron 表

这里的 5 个星号就代表的时间和日期:

第一个*星号代表个小时的第几分钟: minute 范围是从 0-59

第二个*星号代表每天的第几个小时: hour 范围是从 0-23

第三个*星号代表每月的第几个日: day-of-month 范围从 1-31

第四个*星号代表没年的第几个月: month-of-year 范围从 1-12

第五个*星号代表每周的星期几: day-of-week 范围从 0-6, 其中 0 表示星期日

用户名: 也就是执行程序要通过哪个用户来执行, 这个一般可以省略;

命令: 执行的命令和参数。

时程表的格式如下 :

```
f1 f2 f3 f4 f5 program
```

其中 f1 是表示分钟, f2 表示小时, f3 表示一个月份中的第几日, f4 表示月份, f5 表示一个星期中的第几天。program 表示要执行的程序。

当 f1 为 * 时表示每分钟都要执行 program, f2 为 * 时表示每小时都要执行程序, 其余以此类推
当 f1 为 a-b 时表示从第 a 分钟到第 b 分钟这段时间内要执行, f2 为 a-b 时表示从第 a 到第 b 小时都要执行, 其余以此类推

当 f1 为 */n 时表示每 n 分钟个时间间隔执行一次, f2 为 */n 表示每 n 小时个时间间隔执行一次, 其余以此类推

当 f1 为 a, b, c, ... 时表示第 a, b, c, ... 分钟要执行, f2 为 a, b, c, ... 时表示第 a, b, c, ... 个小时要执行, 其余以此类推

使用者也可以将所有的设定先存放在档案 file 中, 用 crontab file 的方式来设定日程表。

例 1: 如果我要表示 9 月 10 日 25 分执行 ls var/spool/cron 任务怎么表示?

```
25 * 10 9 *          ls var/spool/cron
```

由于我没表示小时, 所以就只能里面为每小时

例 2: 我要在每周日, 每分钟执行 wall Hello redking.blog.5lcto.com 这个命令, 时间怎么表示?

```
*/1 * * * * 0 wall Hello redking.blog.5lcto.com
```

***/ 表示每多少分钟执行一次**

例 3: 每晚的 21:30 重启 apache。

```
30 21 * * * /usr/local/etc/rc.d/lighttpd restart
```

例 4: 每月 1、10、22 日的 4 : 45 重启 apache

```
45 4 1,10,22 * * /usr/local/etc/rc.d/lighttpd restart
```

例 5: 每周六、周日的 1 : 10 重启 apache

```
10 1 * * 6,0 /usr/local/etc/rc.d/lighttpd restart
```

例 6: 在每天 18 : 00 至 23 : 00 之间每隔 30 分钟重启 apache

```
0,30 18-23 * * * /usr/local/etc/rc.d/lighttpd restart
```

例 7: 每一小时重启 apache

```
* */1 * * * /usr/local/etc/rc.d/lighttpd restart
```

例 8: 晚上 11 点到早上 7 点之间, 每隔一小时重启 apache

```
* 23-7/1 * * * /usr/local/etc/rc.d/lighttpd restart
```

例 9: 每月的 4 号与每周一到周三的 11 点重启 apache

```
0 11 4 * mon-wed /usr/local/etc/rc.d/lighttpd restart
```

例 10: 一月一号的 4 点重启 apache

```
0 4 1 jan * /usr/local/etc/rc.d/lighttpd restart
```

例 11: 在 12 月内, 每天的早上 6 点到 12 点中, 每隔 3 个小时执行一次 /usr/bin/backup

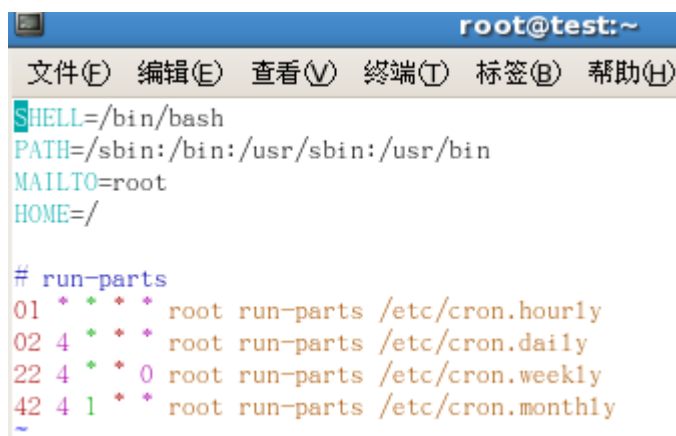
```
0 6-12/3 * 12 * /usr/bin/backup
```

例 12: 每月每天的午夜 0 点 20 分, 2 点 20 分, 4 点 20 分... 执行 echo "haha"

```
20 0-23/2 * * * echo "haha"
```

3、系统级别的计划任务及其扩展anacrontab

这个是系统设置好了, 清理系统垃圾或者是自动执行某些脚本的系统任务, 一般我们做了解就行了, 不要更改配置文件是/etc/crontab



```
root@test:~
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

SHELL: 就是运行计划任务的解释器, 默认是 bash

PATH: 执行命令的环境变量

MAILTO: 计划任务的出发者用户

HOME: 家目录为/

run-parts 是一个脚本, 在/usr/bin/run-parts, 作用是执行一个目录下的所有脚本/程序。

run-parts /etc/cron.hourly 执行目录/etc/cron.hourly/之下的所有脚本/程序。

run-parts 下面就是运行的命令

vim /etc/crontab 与 crontab -e 写入的定时运行的区别?

vim /etc/crontab: 是系统级别定义的 crontab, /etc/crontab 的所有者和组都是 root

crontab -e : 是用户自定义的 crontab, 是所有的用户都可以写入的

两种方法记录的位置不一样，一个在/etc/ 另一个在/var/ 里面。都被 cron 服务调用

如果系统在以上说的时间没有开机怎么办？那么这个脚本不就不能执行了？设计者早就想到了这个问题，所以就有了 cron 服务的扩展, 目的就是为了防止非 24 小时开机的计算机遗漏的守护任务, **anacrontab 就是系统计划任务的扩展文件：在一个指定时间间隔错过后自动执行任务**

格式是这样的：

period delay job-identifier command

period — 命令执行的频率（天数）

delay — 延迟时间（分钟）

job-identifier — 任务的描述，用在 anacron 的消息中，并作为作业时间戳文件的名称，只能包括非空白的字符（除斜线外）。

command — 要执行的命令



```
root@test:~
文件(E) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

1      65      cron.daily      run-parts /etc/cron.daily
7      70      cron.weekly     run-parts /etc/cron.weekly
30     75      cron.monthly    run-parts /etc/cron.monthly
~
```

第一行的意思是：每天开机 65 分钟后就检查 cron.daily 文件是否被执行了，如果今天没有被执行就执行他

第二行的意思是：每隔 7 天开机后 70 分钟检查 cron.weekly 文件是否被执行了，如果一周内没有被执行就执行他

第三行的意思也差不多

下面说说关于 cron 服务的控制，和 at 差不多，就是/etc/cron.deny 这个配置文件来控制，里面写入要禁止使用 cron 用户的名字，一行一个就 OK 了

十七、 VI/VIM编辑器

常用快捷键：

Ctrl+f 向下翻页

Ctrl+b 向上翻页

G 移动到文件最后一行

gg 移动到文件第一行

N+回车 N 为数字，向下移到到 N 行

/关键字 向下寻找关键字

?关键字 向上寻找关键字

从光标向后查找光标所在关键词

* 从光标向前查找光标所在关键词

n 向下重复上一次查找操作

N 与 n 相反，反向重复上一次查找操作

:n1,n2s/关键字 1/关键字 2/g	从第 n1 与 n2 行之间寻找关键字 1，并将关键字 1 替换为关键字 2
:1,\$s/关键字 1/关键字 2/g	从第 1 行到最后一行寻找关键字 1，并将关键字 1 替换为关键字 2
:1,\$s/关键字 1/关键字 2/gc	从第 1 行到最后一行寻找关键字 1，将关键字 1 替换为关键字 2 前会提示用户确认是否替换

dd	删除整行
ndd	n 为数字，删除光标所在向下 n 行。
yy	复制光标所在行
nyy	n 为数字，复制光标所在向下 n 行
p,P	小 p 将复制的数据在光标下一行粘贴，大 P 将复制的数据在光标上一行粘贴
u	撤消前一个操作
Ctrl+r	重做上一个操作
.	将会重复上一个命令
i:	在当前字符的左边插入
I:	在当前行首插入
a:	在当前字符的右边插入
A:	在当前行尾插入
o:	在当前行下面插入一个新行
O:	在当前行上面插入一个新行
:w	保存数据
:wq	保存退出
:q!	不保存退出
:w 文件名	相当于另存为

十八、 压缩打包

linux 下的压缩命令有 tar、gzip、gunzip、bzip2、bunzip2、compress、uncompress、zip、unzip、rar、unrar 等等，压缩后的扩展名有.tar、.gz、.tar.gz、.tgz、.bz2、.tar.bz2、.Z、.tar.Z、.zip、.rar 10 种。

对应关系如下：

- 1、*.tar 用 tar -xvf 解压
- 2、*.gz 用 gzip -d 或者 gunzip 解压
- 3、*.tar.gz 和 *.tgz 用 tar -xzf 解压
- 4、*.bz2 用 bzip2 -d 或者用 bunzip2 解压
- 5、*.tar.bz2 用 tar -xjf 解压
- 6、*.Z 用 uncompress 解压
- 7、*.tar.Z 用 tar -xZf 解压
- 8、*.rar 用 unrar e 解压
- 9、*.zip 用 unzip 解压

在介绍压缩文件之前呢，首先要弄清两个概念：打包和压缩。打包是指将一大堆文件或目录什么的变成一个总的文件，压缩则是将一个大的文件通过一些压缩算法变成一个小文件。为什么要区分这两个概念呢？其实这源于 Linux 中的很多压缩程序只能针对一个文件进行压缩，这样当你想要压缩一大堆文件时，你就得先借助另它的工具将这一大堆文件先打成一个包，然后再就原来的压缩程序进行压缩。

Gzip/zcat

Bzip2/bzcat

Tar

Linux 下最常用的打包程序就是 tar 了，使用 tar 程序打出来的包我们常称为 tar 包，tar 包文件的命令通常都是以 .tar 结尾的。生成 tar 包后，就可以用其它的程序来进行压缩了，所以首先就来讲讲 tar 命令的基本用法：

tar 命令的选项有很多(用 man tar 可以查看到)，但常用的就那么几个选项，下面来举例说明一下：

tar -cf all.tar *.jpg 这条命令是将所有 .jpg 的文件打成一个名为 all.tar 的包。-c 是表示产生新的包，-f 指定包的文件名。

tar -rf all.tar *.gif 这条命令是将所有 .gif 的文件增加到 all.tar 的包里面去。-r 是表示增加文件的意思。

tar -uf all.tar logo.gif 这条命令是更新原来 tar 包 all.tar 中 logo.gif 文件，-u 是表示更新文件的意思。

tar -tf all.tar 这条命令是列出 all.tar 包中所有文件，-t 是列出文件的意思

tar -xf all.tar 这条命令是解出 all.tar 包中所有文件，-t 是解开的意思

以上就是 tar 的最基本的用法。为了方便用户在打包解包的同时可以压缩或解压文件，tar 提供了一种特殊的功能。这就是 tar 可以在打包或解包的同时调用其它的压缩程序，比如调用 gzip、bzip2 等。

1) tar 调用 gzip

gzip 是 GNU 组织开发的一个压缩程序，.gz 结尾的文件就是 gzip 压缩的结果。与 gzip 相对的解压程序是 gunzip。tar 中使用 -z 这个参数来调用 gzip。下面来举例说明一下：

tar -czf all.tar.gz *.jpg 这条命令是将所有 .jpg 的文件打成一个 tar 包，并且将其用 gzip 压缩，生成一个 gzip 压缩过的包，包名为 all.tar.gz

tar -xzf all.tar.gz 这条命令是将上面产生的包解开。

2) tar 调用 bzip2

bzip2 是一个压缩能力更强的压缩程序，.bz2 结尾的文件就是 bzip2 压缩的结果。与 bzip2 相对的解压程序是 bunzip2。tar 中使用 -j 这个参数来调用 bzip2。下面来举例说明一下：

tar -cjf all.tar.bz2 *.jpg 这条命令是将所有 .jpg 的文件打成一个 tar 包，并且将其用 bzip2 压缩，生成一个 bzip2 压缩过的包，包名为 all.tar.bz2

tar -xjf all.tar.bz2 这条命令是将上面产生的包解开。

下面对于 tar 系列的压缩文件作一个小结：

1)对于 .tar 结尾的文件

tar -xf all.tar

2)对于 .gz 结尾的文件

gzip -d all.gz

gunzip all.gz

3)对于 .tgz 或 .tar.gz 结尾的文件

tar -xzf all.tar.gz

tar -xzf all.tgz

4)对于 .bz2 结尾的文件

bzip2 -d all.bz2

bunzip2 all.bz2

5)对于 tar.bz2 结尾的文件

tar -xjf all.tar.bz2

6)对于 .Z 结尾的文件

```
uncompress all.Z
7)对于.tar.Z 结尾的文件
tar -xZf all.tar.z
```

Cpio

Unzip: 解压 zip

Gnuzip: 解压 bz2

十九、性能优化

- 1、设置文件夹打开方式
- 2、设置屏幕保护时间
- 3、解除上网限制
- 4、

二十、常见问题

部分网站无法访问问题的解决

CentOS 5 内核对 TCP 的读缓冲区大小有缺省设置，缺省为：net.ipv4.tcp_rmem = 4096 87380 4194304
解决办法就是将最后一个数字改小一点，具体操作就是在文件/etc/sysctl.conf 中添加一行：

```
net.ipv4.tcp_rmem = 4096 87380 174760
```

然后保存

重新启动网络 service network restart,就 OK 了，如果还是部分网站上不去，可以检查/etc/sysctl.conf 文件是否和下面相同

```
net.ipv4.ip_local_port_range = 1024 65536
net.core.rmem_max=174760
net.core.wmem_max=16777216
net.ipv4.tcp_rmem=4096 87380 174760
net.ipv4.tcp_wmem=4096 65536 16777216
net.ipv4.tcp_fin_timeout = 15
net.ipv4.tcp_keepalive_time = 600
net.ipv4.tcp_tw_recycle = 1
net.core.netdev_max_backlog = 30000
net.ipv4.tcp_no_metrics_save=1
net.core.somaxconn = 262144
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_orphans = 8000
net.ipv4.tcp_max_syn_backlog = 8000
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2
```

net.ipv4.tcp_wmem=4096 65536 16777216 ：为自动调优定义每个 socket 使用的内存。第一个值 4096 是为 socket 的发送缓冲区分配的最少字节数。第二个值 65536 是默认值（该值会被 wmem_default 覆盖），缓冲区在系统负载不重的情况下可以增长到这个值。第三个值 16777216 是发送缓冲区空间的最大字节数（该值会被 wmem_max 覆盖）

net.ipv4.tcp_rmem=4096 87380 174760: 与 tcp_wmem 类似, 不过它表示的是为自动调优所使用的接收缓冲区的值。

net.core.rmem_max = 25165824 #定义最大的 TCP/IP 栈的接收窗口大小

net.core.rmem_default = 25165824 #定义默认的 TCP/IP 栈的接收窗口大小

net.core.wmem_max = 25165824 #定义最大的 TCP/IP 栈的发送窗口大小

net.core.wmem_default = 65536 #定义默认的 TCP/IP 栈的发送窗口大小

net.ipv4.tcp_sack=1 #启用有选择的应答 (Selective Acknowledgment), 这可以通过有选择地应答乱序接收到的报文来提高性能(这样可以使发送者只发送丢失的报文段); (对于广域网通信来说) 这个选项应该启用, 但是这会增加对 CPU 的占用。

net.ipv4.tcp_window_scaling = 1 #启用 RFC1323 定义, 支持超过 64K 窗口

net.ipv4.tcp_fack=1 #启用转发应答 (Forward Acknowledgment), 这可以进行有选择应答 (SACK) 从而减少拥塞情况的发生; 这个选项也应该启用。

net.ipv4.tcp_mem 24576 32768 49152 确定 TCP 栈应该如何反映内存使用; 每个值的单位都是内存页 (通常是 4KB)。第一个值是内存使用的下限。第二个值是内存压力模式开始对缓冲区使用应用压力的上限。第三个值是内存上限。在这个层次上可以将报文丢弃, 从而减少对内存的使用。对于较大的 BDP 可以增大这些值 (但是要记住, 其单位是内存页, 而不是字节)。

Centos5 无法连接无线网络

系统->管理->服务器设置->服务, 将 NetworkManager 选项勾选, 点击重启服务。然后就可以看到右上角已经有了网络连接。

Linux远程管理Windows程序Rdesktop详解

#rpm -q rdesktop //查找是否已经安装

#yum install rdesktop //使用 yum 安装

rdesktop 使用简单, windows 也不和装什么服务端, 是要把远程桌面共享打开就行了
具体使用方法要先打开终端, 然后输入以下命令:

rdesktop -u yourname -p password -g 1024*720 192.168.0.2

rdesktop 为使用远程桌面连接的命令;

-u 用户名, yourname 处为目标客户端的用户名;

-p 客户端用户的密码;

-g 指定使用屏幕大小-g 800*600+0+0 这个 '+0' 就是, 就是你这个窗口的在你 linux 上出现的位置;
192.168.0.1 目标客户端的 IP 地址

实例:

[root@Centos5 ~]# rdesktop -u aixi -p d337448 -r clipboard:PRIMARYCLIPBOARD -r disk:centos=/root -r sound:local -z -a 16 10.26.11.72

\$rdesktop 192.168.1.1 //打开了一个 8 位色彩的,

\$rdesktop -a 16 192.168.1.1 //这个是 16 位色彩的了, 看起来好多了

\$rdesktop -u administrator -p ***** -a 16 192.168.1.1 //都直接登陆了

\$rdesktop -u administrator -p ***** -a 16 -r sound:local 192.168.1.1

加上-r sound:local 可以把声音也搞过来, -r 的作用挺多的可以重定向许多东西, 看一下帮助就会收获不少了。

-r comport:COM1=/dev/ttyS0 // 将串口 /dev/ttyS0 重定向为 COM1

-r comport:COM1=/dev/ttyS0,COM2=/dev/ttyS1 // 多个串口重定向

-r disk:floppy=/mnt/floppy // 将 /mnt/floppy 重定向为远程共享磁盘 'floppy'

-r disk:floppy=/mnt/floppy,cdrom=/mnt/cdrom,root=/,c=/mnt/c // 多个磁盘重定向

-r clientname= // 为重定向的磁盘设置显示的客户端名称
-r lptport:LPT1=/dev/lp0 // 将并口 /dev/lp0 重定向为 LPT1
-r lptport:LPT1=/dev/lp0,LPT2=/dev/lp1 // 多个并口重定向
-r printer:mydeskjet // 打印机重定向
-r printer:mydeskjet="HP LaserJet IIIP" // 打印机重定向
-r sound:[local|off|remote] // 声音重定向

-r clipboard:PRIMARYCLIPBOARD: 这个一定要加上,要不然不能在主机 Solaris 和服务器 Windows 直接复制粘贴文字了。贴中文也没有问题。

-r disk:sunway=/home/jianjian: 指定主机 Solaris 上的一个目录(/home/jianjian)映射到远程 Windows 上的硬盘(盘符为 sunway), 传送文件就不用再靠 Samba 或者 FTP 了。

-f : 全屏, 退出全屏: ctrl+alt+enter 再次 Ctrl+Alt+Enter 即可再次进入全屏

-D: 不显示标题栏, 配合 -g 能更好地使用屏幕空间了;

-K: 这个选项说明保持窗口管理器的按键组合绑定;

-z: 启动网络数据的压缩, 减少带宽, 局域网没什么作用;

提示: 如果你的本地中文文件名在远程机器上显示为乱码的话, 可能是你没有安装编码转化库, 或者你安装的编码转化库不能正确运行。

Linux远程访问Windows共享目录

```
#mount -o username=用户名 -password=密码 //192.168.0.1/C$ /tmp/samba/
```

```
[root@Centos5 ~]# mount -o username=aixi,password=d337448 //10.26.11.72/d$ /root/aixi/
```

说明: IP 地址 192.168.0.1 为中文名文件所在的主机, 文件位于 C 盘, 该主机的用户名及密码为 linux, /tmp/samba/为本地主机挂载目录。在浏览完成后, 使用以下命令卸载。

```
#umount /tmp/samba/
```

升级或安装程序后无法进入图形界面

报错如下:

Failed to start the X server (your graphical interface). It is likely that it is not set up correctly.

Would you like to view the X server output to diagnose the problem ?

解决办法:

```
#cat /var/log/Xorg.0.log | grep EE    查看报错日志
```

```
#sh NVIDIA                        重新安装显卡驱动
```

参考如下网址:

<http://www.linuxquestions.org/questions/linux-hardware-18/failed-to-start-the-x-server-your-graphical-user-interface-605516/>

Linux自动登陆的设置方法

方法一:

1、设置 GDM

GDM 是 GNOME 显示管理器, 通过设置其配置文件/etc/gdm/custom.conf 可以设置帐号自动登陆。

设置方法如下:

在/etc/gdm/custom.conf 文件中添加以下内容

```
[daemon]
```

```
AutomaticLogin=username
```

```
AutomaticLoginEnable=True
```

其中，username 是要自动登陆的用户名。

说明：username 不能是 root，也就说无法实现 root 的自动登陆。

2、设置 prefdm

其中，/etc/inittab 文件的最后一行，该行命令的作用是启动 X Windows，而/etc/X11/prefdm 就是具体实现启动 X Windows 的脚本。

在/etc/X11/prefdm 中添加启动 X Windows 的命令，并退出。

```
/usr/bin/startx
```

```
exit 1
```

说明：

(1)这两行代码一定要在

```
[ -n "$preferred" ] && exec $preferred "$@" >/dev/null 2>&1 </dev/null
```

代码之前。

(2)该方法自动以 root 登陆，是因为运行到/etc/X11/prefdm 时，是 root 身份。

3、在 rc.local 中启动 X Windows

在/etc/rc.local 中添加启动 X Windows 的命令

```
/usr/bin/startx
```

说明：该方法自动以 root 登陆，是因为运行到/etc/rc.local 时，是 root 身份。

以上做完以上的操作就可以实现 Xwindow 的自动登录

方法二：

首先配置自动登录命令行界面

修改/etc/inittab 将 1:2345:respawn:/sbin/mingetty tty1 更改为 1:2345:respawn:/sbin/mingetty tty2

```
--autologin aixi
```

再将/etc/inittab 修改为启动到字符界面：id:3:initdefault

```
#init q
```

使配置生效

这样就可以开机自动启动到命令行界面，如果想自动启动到图形界面，其实在此基础上修改如下：

在/etc/rc.local 中添加启动 X Windows 的命令

```
/usr/bin/startx
```

这样就可以自动启动到命令行，命令行又自动运行 startx 启动图形界面。这是最简单的一种方式。以上在 Centos5.7 版本中测试通过。