

f force 强制 ! 强制 mandatory 强制 optional 可选 通配符 * 来代表每个目录。
builtin 内置命令 keyword 关键字 colon 冒号 disk 磁盘 memory 内存
dash 破折号 slash 斜线 exxx 有效的xxx processes 进程
daemon 守护进程 status 状态
\转义符 \ 回车 为换行 prefix 前缀(基础路径) suffix 后缀 CARRIER 载波
super 键 微软的WIN键
column -t 不管什么都可以用 用管道规整一下输出 自动对齐列

type 查看命令类型，例：该命令是alias，还是内置命令，还是某个文件 -a 列出该命令的所有类型
file 查看的是文件类型，例：文本文件，二进制文件，管道文件，设备文件，链接文件
stat 查看的为文件的属性，例：文件的名称，大小，权限，所有者，atime，ctime等

Install or upgrade an existing system 安装或升级现有的系统
install system with basic video driver 安装过程中采用基本的显卡驱动
Rescue installed system 进入系统修复模式
Boot from local drive 退出安装从硬盘启动
Memory test 内存检测

Linux系统的启动过程，其过程可以分为5个阶段：

- 内核的引导。
- 运行 init。 (init 进程是系统所有进程的起点 init 程序首先是需要读取配置文件 centos7 /usr/lib/systemd/system、 /etc/systemd/system)
- 系统初始化。
- 建立终端。
- 用户登录系统。

Linux系统有7个运行级别(runlevel)：

- 运行级别0：系统停机状态，系统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：多用户状态(没有NFS)
- 运行级别3：完全的多用户状态(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, 登陆后进入图形GUI模式
- 运行级别6: 系统正常关闭并重启, 默认运行级别不能设为6, 否则不能正常启动

关机 `shutdown (init 0/shutdown -h now/poweroff)` 重启 `reboot (init 6/shutdown -r now)`

正确的关机流程为: `sync > shutdown > reboot > halt`

`shutdown` 关机指令, 你可以`man shutdown` 来看一下帮助文档。例如你可以运行如下命令关机:

`shutdown -h 10 ‘This server will shutdown after 10 mins’` 这个命令告诉大家, 计算机将在10分钟后关机, 并且会显示在登陆用户的当前屏幕中。

`shutdown -h now` 立马关机

`shutdown -h 20:25` 系统会在今天20:25关机

`shutdown -h +10` 十分钟后关机

`shutdown -r now` 系统立马重启

`shutdown -r +10` 系统十分钟后重启

`reboot` 就是重启, 等同于 `shutdown -r now`

`halt` 关闭系统, 等同于`shutdown -h now` 和 `poweroff`

`/bin:`

`bin` 是 `Binaries` (二进制文件) 的缩写, 这个目录存放着最经常使用的命令。

`/boot:`

这里存放的是启动 `Linux` 时使用的一些核心文件, 包括一些连接文件以及镜像文件。

`/dev :`

`dev` 是 `Device`(设备) 的缩写, 该目录下存放的是 `Linux` 的外部设备, 在 `Linux` 中访问设备的方式和访问文件的方式是相同的。

`/etc:`

`etc` 是 `Etcetera`(等等) 的缩写, 这个目录用来存放所有的系统管理所需要的配置文件和子目录。

`/home:`

用户的主目录, 在 `Linux` 中, 每个用户都有一个自己的目录, 一般该目录名是以用户的账号命名的。

`/lib:`

`lib` 是 `Library`(库) 的缩写这个目录里存放着系统最基本的动态连接共享库, 其作用类似于 `windows` 里的 `DLL` 文件。几乎所有的应用程序都需要用到这些共享库。

`/lost+found:`

这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。

/media:

linux 系统会自动识别一些设备，例如U盘、光驱等等，当识别后，Linux 会把识别的设备挂载到这个目录下。

/mnt:

系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将光驱挂载在 /mnt/ 上，然后进入该目录就可以查看光驱里的内容了。

/opt:

opt 是 optional(可选) 的缩写，这是给主机额外安装软件所摆放的目录。比如你安装一个ORACLE 数据库则就可以放到这个目录下。默认是空的。

/proc:

proc 是 Processes(进程) 的缩写，/proc 是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，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。

这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件，比如可以通过下面的命令来屏蔽主机的ping命令，使别人无法ping你的机器：

```
禁ping      echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

/root:

该目录为系统管理员，也称作超级权限者的用户主目录。

/sbin:

s 就是 Super User 的意思，是 Superuser Binaries (超级用户的二进制文件) 的缩写，这里存放的是系统管理员使用的系统管理程序。

/selinux:

这个目录是 Redhat/Centos 所特有的目录，selinux 是一个安全机制，类似于 windows 的防火墙，但是这套机制比较复杂，这个目录就是存放selinux相关的文件的。

/srv:

该目录存放一些服务启动之后需要提取的数据。

/sys:

这是 Linux2.6 内核的一个很大的变化。该目录下安装了 2.6 内核中新出现的一个文件系统 sysfs 。

sysfs 文件系统集成了下面3种文件系统的信息：针对进程信息的 proc 文件系统、针对设备的 devfs 文件系统以及针对伪终端的 devpts 文件系统。

该文件系统是内核设备树的一个直观反映。

当一个内核对象被创建的时候，对应的文件和目录也在内核对象子系统中被创建。

/tmp:

tmp 是 temporary(临时) 的缩写这个目录是用来存放一些临时文件的。

/usr:

usr 是 unix shared resources(共享资源) 的缩写，这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似于 windows 下的 C:\windows 目录。

/usr/local 类似于 windows C盘下的 program files 目录。

/usr/bin:

系统用户使用的应用程序。

/usr/sbin:

超级用户使用的比较高级的管理程序和系统守护程序。

/usr/src:

内核源代码默认的放置目录。

/var:

var 是 variable(变量) 的缩写，这个目录中存放着在不断扩充着的东西，我们习惯将那些经常被修改的目录放在这个目录下。包括各种日志文件。

/run:

是一个临时文件系统，存储系统启动以来的信息。当系统重启时，这个目录下的文件应该被删掉或清除。如果你的系统上有 /var/run 目录，应该让它指向 run。

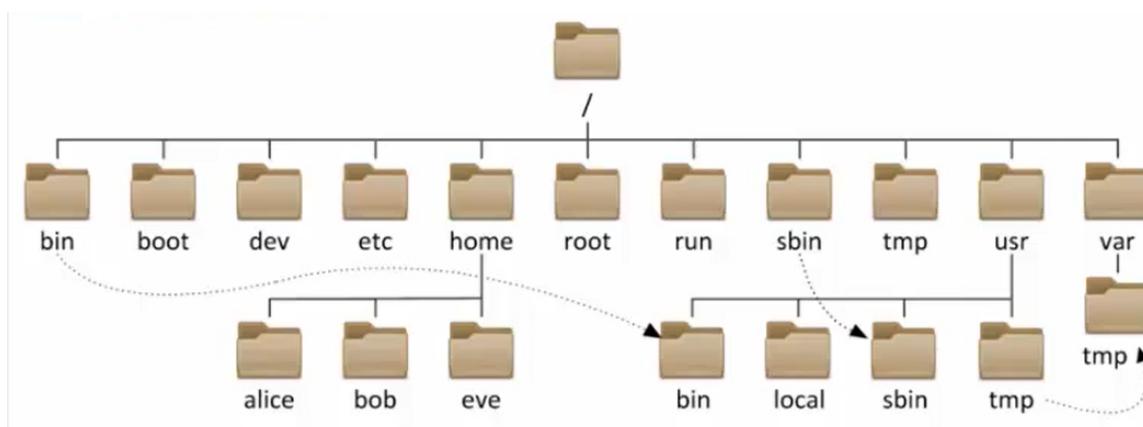
在 Linux 系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

/etc: 上边也提到了，这个是系统中的配置文件，如果你更改了该目录下的某个文件可能会导致系统不能启动。

/bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录，比如 ls 就是在 /bin/ls 目录下的。

值得提出的是，/bin, /usr/bin 是给系统用户使用的指令（除root外的通用用户），而 /sbin, /usr/sbin 则是给 root 使用的指令。

/var: 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在 /var/log 目录下，另外 mail 的预设放置也是在这里。



boot	存放的系统启动相关的文件，例如kernel,grub(引导装载程序)
	[root@tianyun ~]# ls /boot
	config-3.10.0-514.el7.x86_64
	grub
	grub2[
	initramfs-0-rescue-7dd7296d4eaa44dcb0e4ec6104efa2bc.img
	initramfs-3.10.0-514.el7.x86_64.img
	initrd-plymouth.img
	symvers-3.10.0-514.el7.x86_64.gz
	System.map-3.10.0-514.el7.x86_64
	vmlinuz-0-rescue-7dd7296d4eaa44dcb0e4ec6104efa2bc
	vmlinuz-3.10.0-514.el7.x86_64 ← kernel 内核
etc	配置文件 (系统相关如网络/etc/sysconfig/network，应用相关配置文件如/etc/ssh/sshd_config...)
lib	库文件Glibc
lib64	库文件Glibc
tmp	临时文件(全局可写：进程产生的临时文件)
var	存放的是一些变化文件，比如数据库，日志，邮件....
	mysql: /var/lib/mysql
	vsftpd: /var/ftp
	mail: /var/spool/mail
	cron: /var/spool/cron
	log: /var/log
	临时文件: /var/tmp(进程产生的临时文件)
bin	普通用户使用的命令 /bin/ls, /bin/date
sbin	管理员使用的命令 /sbin/service
	[root@tianyun ~]# which ls
	alias ls='ls --color=auto'
	/usr/bin/ls
	[root@tianyun ~]#
	[root@tianyun ~]# which useradd
	/usr/sbin/useradd
dev	设备文件 /dev/sda,/dev/sda1,/dev/tty1,/dev/tty2,/dev/pts/1, /dev/zero, /dev/null, /dev/random
root	root用户的HOME
home	存储普通用户家目录
proc	虚拟的文件系统，反映出来的是内核，进程信息或实时状态
usr	系统文件，相当于C:\Windows
	/usr/local 软件安装的目录，相当于C:\Program
	/usr/bin 普通用户使用的应用程序
	/usr/sbin 管理员使用的应用程序
	/usr/lib 库文件Glibc
	/usr/lib64 库文件Glibc
	crw-rw-rw- 1 root root 1, 3 7月 24 06:28 /dev/null 空设备，类似回收站
	crw-rw-rw- 1 root root 1, 8 7月 24 06:28 /dev/random 产生随机数
	crw-rw-rw- 1 root root 1, 5 7月 24 06:28 /dev/zero 零设备

忘记密码：

重启---->启动界面在3秒内按下回车---->按e---->在第二行最后边输入 single，有一个空格。具体方法为按向右尖头移动到第二行，按"e"进入编辑模式---->在后边加上single 回车---->最后按"b"启动，启动后就进入了单用户模式了--->此时已经进入到单用户模式了，你可以更改root密码了。更密码的命令为 passwd

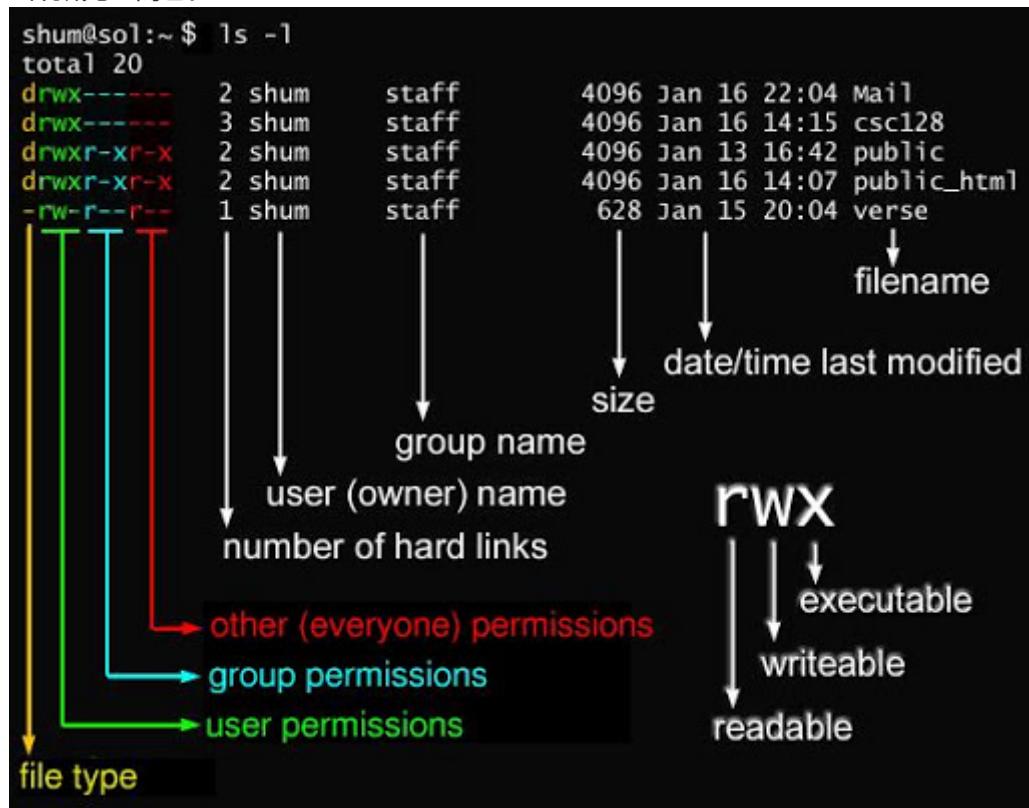
ssh 端口为22 SSH 为建立在应用层和传输层基础上的安全协议。

```
dr-xr-xr-x  2 root root 4096 Dec 14 2012 bin
```

在 Linux 中第一个字符代表这个文件是目录、文件或链接文件等等。

- 当为 d 则是目录
- 当为 - 则是文件;
- 若是 l 则表示为链接文档(link file);
- 若是 b 则表示为装置文件里面的可供储存的接口设备(可随机存取装置);
- 若是 c 则表示为装置文件里面的串行端口设备，例如键盘、鼠标(一次性读取装置)。

接下来的字符中，以三个为一组，且均为 rwx 的三个参数的组合。其中，r 代表可读(read)、w 代表可写(write)、x 代表可执行(execute)。要注意的是，这三个权限的位置不会改变，如果没有权限，就会出现减号 - 而已。



文件类型	属主权限	属组权限	其他用户权限
0	1 2 3	4 5 6	7 8 9
d 目录文件	rwx 读 写 执行	r-X 读 写 执行	r-X 读 写 执行

从左至右用 0-9 这些数字来表示。

第 0 位确定文件类型，第 1-3 位确定属主（该文件的所有者）拥有该文件的权限。

第4-6位确定属组（所有者的同组用户）拥有该文件的权限，第7-9位确定其他用户拥有该文件的权限。

其中，第 1、4、7 位表示读权限，如果用 r 字符表示，则有读权限，如果用 - 字符表示，则没有读权限；

第 2、5、8 位表示写权限，如果用 w 字符表示，则有写权限，如果用 - 字符表示没有写权限；第 3、6、9 位表示可执行权限，如果用 x 字符表示，则有执行权限，如果用 - 字符表示，则没有执行权限。

权限值：r: 4, w: 2, x: 1. 最高值为 777.

ugo设置基本权限:r(读),w(写),x(执行)

acl设置基本权限：r(读),w(写),x(执行)

改变文件的所有者与所属组：

以yu用户 chao所属组 file1文件为例

在 Linux 中我们通常使用以下两个命令来修改文件或目录的所属用户与权限：

- chown (change ownerp) : 修改所属用户与组。
- chmod (change mode) : 修改某个文件的访问模式，修改用户的权限。

chown yu file1 //更改file1文件的所有者为yu用户，只改所有者

chown yu.chao file1 //把file1文件的所有者改为yu, 所属组改为chao, 同时更改所有者和所属组, 所有者和所属组间以.或:隔开

chown chao file1 //会报错，因为不存在chao这个用户，需要加.区分用户和组

chown .chao file1 //将file1文件的所属组更改为chao 只改所属组

chown -R yu.chao 目录 //将目录及目录下所有文件的所有者和所属组都改为yu, chao

-R, --recursive 递归处理所有的文件及子目录

chgrp 只能更改所属组

chgrp chao file1 //将file1文件的所属组更改为chao, 相当于 chown .chao file1

user:	用户	u
group:	组	g
others:	其他	o
all :	全部身份	a

```
chmod      u,g,o,a  + (写入), - (除去), = (设定)      r(4), w(2), x(1)
# touch test1 // 创建 test1 文件
# ls -al test1 // 查看 test1 默认权限
-rw-r--r-- 1 root root 0 Nov 15 10:32 test1
# chmod u=rwx,g=rx,o=r  test1 // 修改 test1 权限
# ls -al test1
-rwxr-xr-- 1 root root 0 Nov 15 10:32 test1

# chmod u-x,g+w,o+w  test1
-rw-rwxrw- 1 root root 0 Nov 15 10:32 test1 //增减权限
```

```

# chmod a=rwx test1 //为所有人添加rwx权限
# chmod 777 test1 //和上边效果一样

# chmod a=- test1 //所有人都没有权限
# chmod 000 test1 //所有人都没有权限

# chmod 644 test1 //6为所有者的权限,4为属组的权限,4其他人的权限

7 = 007 77 = 077 777 = 777
例:# chmod 7 test1 //所有者和所属组的权限为0,其他人的权限为rwx(7),一般用用3位数设置权限最保险

```

权限	对文件的影响	对目录的影响
r (读取)	可以读取文件的内容	可以列出目录的内容 (文件名)
w (写入)	可以更改文件的内容	可以创建或删除目录中的任一文件
x (可执行)	可以作为命令执行文件	可以访问目录的内容 (取决于目录中文件的权限)

对目录没有写权限,即使对目录下的文件有r,w,x权限,其也不能删除目录下的任何文件,对目录有w权限,对文件没有任何权限,用户也可以删除目录下的任何文件和创建(删除文件与文件权限无关只与其父目录有关)。

root和文件所有者除外 我的附庸的附庸不是我的附庸。 目录1-->目录2-->文件1 目录1的w权限删除不了目录2和文件1,只有目录2的w权限才能删除文件1,若目录1下有一个文件2,目录1能删除文件2

注意:对于文件来说执行(x)的权限得小心给予

对于目录来说写(w)权限得小心给予

acl (access control lists) 设置文件基本权限:

fac1文件访问控制列表 f 文件

getfac1 查看文件的acl权限

setfac1 设置文件的acl权限 -m 修改文件的acl权限

例:setfac1 -m u(或user):yu:r /home/test.txt //修改用户yu对test.txt文件的权限为rw

-x 删除XXX的acl权限 -b 删除全部acl权限

setfac1 -x 用户

setfac1 -b /home/wo.txt //将wo.txt的acl权限都删除

删除前设置的acl权限

删除前 -rw-rw-r--+ 1 root root 0 Sep 24 21:13 /home/wo.txt

删除后 -rw----r-- 1 root root 0 Sep 24 21:13 /home/wo.txt

一旦文件权限后边多个+代表着文件有了新的acl权限 mask 掩码,面具

11 查看文件权限后 有 + 要用getfac1查看其的acl权限

mask 用于临时降低用户或组(除属主或其他人)的权限,决定了有命名的用户和组的最高权限,除了所有者和其他人都受mask的影响 //为了方便管理文件权限,其他人的权限置为空,因为将mask权限置为空的时候,有命名的用户或组会有其他人的权限

例:setfac1 -m mask::-- /home/file1.txt(置空文件的路径)

mask一经设置之后，任何重新设置acl权限，那么mask就会失效。所以说mask是临时的降低权限。

mask的权限始终会跟随权限最多那个用户或组，设置了mask的权限其他用户或组即是有权限也相当于没有例:[root@yu home]# setfacl -m u:yu:rwx,u:www:rw /home/wo.txt//为用户www和yu设置acl权限

```
[root@yu home]# getfacl /home/wo.txt //查看wo.txt文件的acl权限
getfacl: Removing leading '/' from absolute path names
# file: home/wo.txt
# owner: root
# group: root
user::rw-
user:www:rw-
user:yu:rwx
group::r--
mask::rwx
other::r--
```

```
[root@yu home]# setfacl -m mask::--- /home/wo.txt //将mask的权限设置为空
[root@yu home]# getfacl /home/wo.txt
getfacl: Removing leading '/' from absolute path names
# file: home/wo.txt
# owner: root
# group: root
user::rw-
user:www:rw-      #effective:--- //effective 有效的权限
user:yu:rwx        #effective:---
group::r--         #effective:---
mask:---
other::r--         //因为mask的权限为---所以即使用户www有rw权限和用户yu有rwx权限,但其实实际上使用它们是没有权限的,有other(其他人)的权限
```

default:继承(默认) 继承是以后的问题 要求:希望yu用户能够对/home以及以后在/home下新建的目录和文件有读、写、执行权限 /home为示例

思路: setfacl -m u:yu:rwx /home //赋予yu对/home的读、写、执行权限
-R 递归继承
setfacl -m d(或default):u:yu:rwx /home //赋予yu对以后在/home下新建的目录和文件有读、写、执行权限(使yu用户的权限继承)

```
[root@yu /]# mkdir 111 //创建一个111目录
[root@yu /]# setfacl -m d:u:yu:rwx /111 //给111目录赋予继承属性
[root@yu /]# getfacl /111 //查看111目录的acl权限
getfacl: Removing leading '/' from absolute path names
# file: 111
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
default:user::rwx    //default 继承
default:user:yu:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

```
[root@iz2jsvs2ddurawz /]# mkdir /111/222 //在目录111下创建目录222
[root@iz2jsvs2ddurawz /]# getfacl /111/222 //查看222的权限
```

```
getfacl: Removing leading '/' from absolute path names
# file: 111/222
# owner: root
# group: root
user::rwx
user:yu:rwx
group::r-x
mask::rwx
other::r-x
default:user::rwx
default:user:yu:rwx //新建的222继承了目录111的权限
default:group::r-x
default:mask::rwx
default:other::r-x
```

umask 显示或设定文件的模式掩码。

Linux umask命令指定在建立文件时预设的权限掩码。

当我们登录系统之后创建一个文件总是有一个默认权限的，那么这个权限是怎么来的呢？这就是umask干的事情。umask设置了用户创建文件的默认权限，它与chmod的效果刚好相反，umask设置的是权限“补码”，而chmod设置的是文件权限码。一般在/etc/profile、/.bash_profile或家目录/.profile中设置umask值。

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。

语法：

umask [-S] [权限掩码]

执行上面的指令后，输出信息如下

```
$ umask #获取当前权限掩码
```

```
$ mkdir test1 #创建目录
```

```
$ ls -d -l test1/ #显示目录的详细信息
```

```
drwxr-xr-x 2 rootlocal rootlocal 4096 2011-9-19 21:46 test1/
```

在上面的输出信息中，“drwxr-xr-x”=“777-022=755”

文件一开始创建是不可能有执行权限的。必须后期赋予。

文件权限管理之：进程umask

进程 新建文件、目录的默认权限会受到umask的影响，umask表示要减掉的权限

shell (vim, touch)	=====umask===== >	新文件或目录权限
vsftpd	=====umask===== >	新文件或目录权限
samba	=====umask===== >	新文件或目录权限
useradd	=====umask===== >	用户HOME

示例3：修改shell umask值（永久 建议不要）

```
[root@tianyun ~]# vim /etc/profile
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi
[root@tianyun ~]# source /etc/profile //立即在当前shell中生效
```

示例4：通过umask决定新建用户HOME目录的权限

```
[root@tianyun ~]# vim /etc/login.defs
UMASK      077
[root@tianyun ~]# useradd gougou
[root@tianyun ~]# ll -d /home/gougou/
drwx-----. 4 gougou gougou 4096 3月 11 19:50 /home/gougou/

[root@tianyun ~]# vim /etc/login.defs
UMASK      000
[root@tianyun ~]# useradd yangyang
[root@tianyun ~]# ll -d /home/yangyang/
drwxrwxrwx. 4 yangyang yangyang 4096 3月 11 19:53 /home/yangyang
```

示例5：例如vsftpd进程 /etc/vsftpd/vsftpd.conf 【了解】

```
anon_umask = 007
local_umask = 000
```

示例1：在shell进程中创建文件

```
[root@tianyun ~]# umask //查看当前用户的umask权限
0022
[root@tianyun ~]# touch file800
[root@tianyun ~]# mkdir dir800
[root@tianyun ~]# ll -d dir800 file800
drwxr-xr-x. 2 root root 4096 3月 11 19:40 dir800
-rw-r--r--. 1 root root 0 3月 11 19:40 file800
```

示例2：修改shell umask值（临时）

```
[root@tianyun ~]# umask 000
[root@tianyun ~]# mkdir dir900
[root@tianyun ~]# touch file900
[root@tianyun ~]# ll -d dir900 file900
drwxrwxrwx. 2 root root 4096 3月 11 19:44 dir900
-rw-rw-rw-. 1 root root 0 3月 11 19:44 file900
```

高级权限(特殊权限) uid,sgid,sticky

```
suid 4      sgid 2      sticky 1
chmod u+s ni //ni文件 wo目录
chmod g+s ni //一般sgid是不会赋给文件的，不推荐只了解即可
chmod g+s wo
chmod o+t wo
```

s(setuid)： 任何人在使用该程序的时候其身份都是该文件的所有者(一般都为root)(针对文件)

例：

修改密码的时候 which passwd passwd命令位于 /usr/bin/passwd

而调用的是 /etc/shadow

```
[root@iz2jsvs2ddurawz ~]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 34928 May 11 2019 /usr/bin/passwd //s setuid
```

sgid 新建文件继承目录属组(针对目录)
在文件夹上加上**sgid**,在该文件夹下的新建的文件的属组会继承属组
例:

```
[root@iz2jsvs2ddurawZ home]# mkdir wo //创建文件夹wo
[root@iz2jsvs2ddurawZ home]# ll -d wo //查看wo的权限
drwxr-xr-x 2 root root 6 Sep 25 21:12 wo
[root@iz2jsvs2ddurawZ home]# chgrp yu wo //更改wo目录的所属组
[root@iz2jsvs2ddurawZ home]# ll -d wo //查看文件权限的变化
drwxr-xr-x 2 root yu 6 Sep 25 21:12 wo
[root@iz2jsvs2ddurawZ home]# chmod g+s wo //给wo目录的组增加s
[root@iz2jsvs2ddurawZ home]# ll -d wo
drwxr-sr-x 2 root yu 6 Sep 25 21:12 wo

[root@iz2jsvs2ddurawZ wo]# ll file
-rw-r--r-- 1 root yu 0 Sep 25 21:18 file //新建的file继承了父目录wo的属组
```

sticky t(粘滞位) (针对目录) 用户只能删除自己的文件 加入这个权限后 只有**root**,文件的所有者和目录的所有者才能删除文件

/tmp /var/tmp 临时目录
以下文件的权限必须是1777否则会导致程序不能正常运行。

```
[root@localhost ~]# ll -d /tmp
drwxrwxrwt. 39 root root 4096 9月 27 00:32 /tmp
[root@localhost ~]# ll -d /var/tmp
drwxrwxrwt. 2 root root 4096 9月 20 05:12 /var/tmp
权限值为1777 1为t的权限 此文件全局可写
```

```
[root@localhost home]# chmod 7777 wo
[root@localhost home]# ll -d wo
drwsrwsrwt. 2 root root 4096 9月 27 00:46 wo
[root@localhost home]# chmod a-x wo
[root@localhost home]# ll -d wo
drwsrwSrwt. 2 root root 4096 9月 27 00:46 o //没有x执行权限就会变成大写
SST(suid,sgid,sticky)
```

文件属性

文件属性高于权限

lsattr 查看文件属性 **chattr** 赋予文件属性

chattr **chattr - change file attributes on a Linux file system**
用法: **chattr (参数) (版本) 文件** //文件为必选项 赋予属性 **+-=**

chattr +a 只能追加内容,不能删除,写等权限 一般用于日志文件的限制

chattr +i (immutable 不可改变) 加上这个属性文件什么操作都不允许//可以给**passwd**等保密文件赋予

例:

```
[root@localhost home]# chattr +a ni
[root@localhost home]# lsattr ni
-----a-----e- ni //e属性是ext4的文件系统属性
[root@localhost home]# vi ni
ni" E212: can't open file for writing //不允许写入, 强制写入也不行
[root@localhost home]# cat ni
[root@localhost home]# rm -rf ni
rm: 无法删除"ni": 不允许的操作
```

```
[root@localhost home]# echo 1111 >> ni //追加内容 1111 只有追加才能写入  
//echo > 往进写 >>追加内容  
[root@localhost home]# cat ni  
1111  
若想删除属性 chattr -(acdeijstuADST) 文件  
//chattr = 文件 这样也可以删除文件属性 centos6不支持  
[root@localhost home]# chattr -a ni  
[root@localhost home]# lsattr ni  
-----e- ni
```

- ls (英文全拼: list files) :列出目录及文件名 -a 全部文件 -d 查看文件本身的信息 (其他的选项为查看文件夹里文件的信息) -l 显示文件属性和权限 -h 以易阅读的方式显示 -r 逆序排序显示 -t 按修改时间排序, 最新的排在前边 -S 以文件的尺寸排序
- ll 长文件显示 -i 显示文件的inode号 (索引号) (每个文件在磁盘中的目录)
- cd(英文全拼: change directory)切换目录 cd - 返回上一次停留的目录 cd / 返回根目录 cd .. 返回上级目录 cd ~ 或 cd 返回家目录
- pwd (英文全拼: print work directory) : 显示目前的目录 -P 显示确实路径, 而非连结 (link) 路径
- mkdir (英文全拼: make directory) : 创建一个新的目录 -m配置文件权限 -p 递归创建
mkdir -m a=rwx 目录名 //a/u/g/o=rwx
- rmdir (英文全拼: remove directory) : 删除一个空的目录 -P 连同上级空目录一起删除
- cp (英文全拼: copy file) : 复制文件或目录
- rm (英文全拼: remove) : 移除文件或目录
 - f : 就是 force 的意思, 忽略不存在的文件, 不会出现警告信息;
 - i : 互动模式, 在删除前会询问使用者是否动作
 - r : 递归删除! 最常用在目录的删除了!
 - v: 详细信息。
- mv (英文全拼: move file) : 移动文件与目录, 或修改文件与目录的名称
 - f : force 强制的意思, 如果目标文件已经存在, 不会询问而直接覆盖;
 - i : 若目标文件(destination) 已经存在时, 就会询问是否覆盖!
 - u : 若目标文件已经存在, 且 source 比较新, 才会升级(update)

你可以使用 *man [命令]* 来查看各个命令的使用文档, 如 : man cp.

Linux文件类型

(目录是一个索引)

Linux不是通过扩展名来区分文件的

查看文件类型的方法:

第一种方法：

ls -l 文件名 //看第一个字符

- 普通文件(例：文本文件，二进制文件，压缩文件，电影，图片等)

directory 目录 d 目录文件（蓝色）

block special 块设备 b 设备文件（块设备）存储设备，U盘/dev/sda,/dev/sda1等

character special 字符设备 c 设备文件（字符设备）打印机，终端/dev/tty1等

socket 套接字文件 s 套接字文件

fifo (name pipe 命名管道) p 管道文件 fifo(first input first output 先进先出)

symbolic link 符号连接 l 链接文件（淡蓝色）（相当于Windows的快捷方式）

第二种方法：

file 命令

file 文件名/文件路径

```
[root@localhost ~]# ll /dev/sda /dev/null /etc/hosts
crw-rw-rw-. 1 root root 1, 3 10月 4 20:56 /dev/null
          1为主设备号，3为从设备号
brw-rw----. 1 root disk 8, 0 10月 4 20:56 /dev/sda
          8为(MAJOR)主设备号，0为(MINOR)从设备号
brw-rw----. 1 root disk 8, 1 10月 4 20:56 /dev/sda1
brw-rw----. 1 root disk 8, 2 10月 4 20:56 /dev/sda2

-rw-r--r--. 1 root root 83 6月 30 2018 /etc/hosts
          83为文件大小
```

设备文件都有主设备号和从设备号，普通文件有文件大小

主设备号相同；表示为同一种设备类型，同时可认为kernel（内核）使用的是相同的驱动（模块）；从设备号：在同一类型设备中的一个序号。

字符设备和块设备的区别：块设备中有缓存。 块设备认为是硬盘就行。

Linux 文件内容查看

Linux下的文件编辑器：gedit（图形化编辑，需要图形化支持），vi, vim, nano。

```
装vim    yum -y install vim-enhanced
```

Linux系统中使用以下命令来查看文件的内容：

```
windows转Linux格式的工具    yum -y install dos2unix
```

使用：dos2unix 需要转的文件 回车 在windows打开Linux文件一般用记事本打开，在Linux里才需要转格式

- cat 由第一行开始显示文件内容 Windows里的换行符为^M\$ Linux里的换行符为\$ Linux里所有文件默认都有一个换行符
 - -A：相当於 -vET 的整合选项，可列出一些特殊字符（包括换行符，制表符）而不是空白而已；
 - -b：列出行号，仅针对非空白行做行号显示，空白行不标行号！

- -E：将结尾的断行字节 \$ 显NL 示出来；
 - -n：列印出行号，连同空白行也会有行号，与 -b 的选项不同；
 - -T：将 [tab] 按键以 \t 显示出来；
 - -v：列出一些看不出来的特殊字符
- tac 从最后一行开始显示，可以看出 tac 是 cat 的倒着写！
- nl 显示的时候，顺道输出行号！
 - -b：指定行号指定的方式，主要有两种：
 - b a：表示不论是否为空行，也同样列出行号(类似 cat -n)；
 - b t：如果有空行，空的那一行不要列出行号(默认值)；
 - -n：列出行号表示的方法，主要有三种：
 - n ln：行号在荧幕的最左方显示；
 - n rn：行号在自己栏位的最右方显示，且不加 0；
 - n rz：行号在自己栏位的最右方显示，且加 0；
 - -w：行号栏位的占用的位数。
- more 一页一页的显示文件内容
 - 空白键 (space)：代表向下翻一页；
 - Enter：代表向下翻『一行』；
 - /字符串：代表在这个显示的内容当中，向下搜寻『字符串』这个关键字；
 - :f：立刻显示出档名以及目前显示的行数；
 - q：代表立刻离开 more，不再显示该文件内容。
 - b 或 [ctrl]-b：代表往回翻页，不过这动作只对文件有用，对管线无用。
- less 与 more 类似，但是比 more 更好的是，他可以往前翻页！
 - 空白键：向下翻动一页；
 - [pagedown]：向下翻动一页；
 - [pageup]：向上翻动一页；
 - /字符串：向下搜寻『字符串』的功能；
 - ?字符串：向上搜寻『字符串』的功能；
 - n：重复前一个搜寻(与 / 或 ? 有关！)
 - N：反向的重复前一个搜寻(与 / 或 ? 有关！)
 - q：离开 less 这个程序；
- head 只看头几行
 - -n：后面接数字，代表显示几行的意思 head -n 3 显示前三行
- tail 只看尾巴几行
 - -n：后面接数字，代表显示几行的意思 tail -n 3 显示末尾三行
 - -f：表示持续侦测后面所接的档名，要等到按下[ctrl]-c才会结束tail的侦测 tail -f 等价于 tailf
- grep 过滤查看文件内容


```
#grep '过滤的内容' 路径 例: grep 'root' /etc/passwd 在passwd里查找包含root的内容
#可以搭配正则表达式使用 例: grep '^root' /etc/passwd 在passwd里查找以root开头的内容
#可以搭配正则表达式使用 例: grep 'root$' /etc/passwd 在passwd里查找以root结尾的内容
#grep 'failure' /var/log/secure 过滤通过非法登录计算机的内容

grep过滤信息时: grep 不等于'grep' 要过滤有空格的信息必须先用引号引起来
grep 在过滤含有-或--的内容时要加"和转义字符\ 例: grep '\ --force' //如果不加系统会认为--force为grep的选项
```

Linux文件时间

```
ls -l 文件名 仅查看的是文件的修改时间      ll 文件 --full-time 最具体的更改时间//ll wO  
--full-time
```

Linux文件的四种时间：(无创建时间)

stat用法: **stat** 文件名 //查看文件的详细属性 (其中包含文件的时间属性)

访问时间(**Access** 最近访问):**atime**,查看内容

修改时间(内容的变动 **Modify** 最近更改):**mtime**,修改内容

改变时间(属性的变动 **Change** 最近改动):**ctime**,文件属性,比如权限等

删除时间:**dtime**,文件被删除的时间

从RHEL开始**relatime**,**atime**延迟修改!必须满足其中一个条件:

1.自上次**atime**修改后,已达到86400秒(1天);2.发生写操作时。

Linux系统用户与组

- 用户与组:
- 1.系统上的每个进程 (运行的程序) 都是为特定的用户所运行的。
 - 2.每个文件是由一个特定的用户所拥有
 - 3.进程访问文件和目录受到用户的限制 (每个进程能访问的文件是由运行该进程的用户所决定的, 用户是不能访问文件的是由其所拥有的进程访问)
 - 4.正在运行的进程能够以何种方式去访问文件和目录是由其所关联的用户决定的

查看当前登录的用户信息:

```
id 命令    id 用户  
uid(用户ID)  gid(组ID)  用户所属的组ID  groups(用户所有组的ID)
```

查看文件的owner(所有者):

```
ll /home/
```

查看运行进程的username:

```
ps aux | less(more)
```

查看用户的邮箱路径 /var/spool/mail

查看家里的用户 /home/

给用户设置密码:

在root下 **passwd** 用户名 **root**可以给任何用户设置密码, 只有**root**为其他用户设置密码

passwd时其后才需要加用户名, 普通用户直接在普通用户时用**passwd**

在用户模式下 **passwd** 普通用户只能给自己设置密码, 而且需要提供原密码

和用户和组相关的文件:

/etc/passwd(里面为用户信息)

```
root:x:0:0:root:/root:/bin/bash
```

用户名:**x**(密码占位符):**uid**:**gid**(主组):描述:目录:**shell** (在之前密码也在这个文件里后来为了安全考虑分割出去了所以有个密码占位符)

/etc/shadow(里面为密码信息)

```
root:$1盐加密结果::0:99999::::
```

\$id\$salt\$encrypted

用户:加密的密码:最后加密的时间(为天数,从1970年开始算):密码最小年龄(密码的存活时间,按天记数):密码最大年龄(密码存活的最长时间,到时候会强制改密):在用户密码将要过期的前几天提醒用户:密码过期后的几天内该用户将被禁用:密码的过期时间(时间到了无改密机会,无法登录):保留项

例：两个密码，同样的密码，同样的算法，加密出的结果就会相同，加盐的话加密出的结果就会不同）

加密的密码=(分三段)加密算法类型盐加密结果

系统约定:centos6 centos7
uid:0 特权用户
uid:1~499 系统用户
uid:500+ 普通用户 uid:1000+ 普通用户

root用户：

UID: 0, 所有权力, root用户权力覆盖文件系统上的普通权限, 安装或删除软件并管理系统文件和目录, 大多数设备只能由root控制(如:硬盘, 存储等)

加密算法\$id:

\$1: MD5

\$5: SHA-256

§6: SHA-512(哈希算法)

组：如果创建一个用户时，未指定任何组(主组或附加组)，系统会创建一个和用户名相同的组作为用户的主组(**Primary Group**)。

用户的主组必须有且只有一个，

/etc/group(里面为组信息)

root:x:0: 组信息
组名:密码占位符:组的qid:组的成员

groupadd 创建用户组 **groupadd -g/gid** 创建用户组并赋予**gid** 一般不需要专门赋予
gid系统会自动给予**gid**

useradd 选项 用户名

useradd 创建用户

新建的用户会从 /etc/skel/ 中将所有文件拷贝到 /home/ 新建用户的目录下 查看 skel 目录文件 ls /etc/skel/ -a

用户目录下没有.bash等文件会出现 -bash-4.2\$ 错误 只需将skel里的文件拷贝到用户目录下就行 cp -rf /etc/skel/.bash* /home/缺bash文件的用户名/

useradd参照文件 /etc/login.defs(关于密码的最长时间,最小长度, 启动何种算法等)
/etc/default/useradd(用户的家目录,用户的shell等)

- `-f comment` 指定一段注释性描述。

- -d 目录 指定用户家目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。
- -g 用户组 指定用户所属的用户组。
- -G 指定用户所属的附加组。
- -s Shell文件 指定用户的登录Shell。
- -u 用户号 指定用户的uid，如果同时有-o选项，则可以重复使用其他用户的标识号。

useradd 用户 -s /sbin/随便起个shell(如:nologin) 那么这个用户将是一个安全用户,有shell无法登录系统

grep 'bash\$' /etc/passwd 过滤以bash结尾的 有bash就有登录

有的用户有shell但无法登录系统，实现管理。仅作为运行进程的用户，访问ftp的用户。

useradd和adduser的区别：

Linux下创建用户时会用到useradd和adduser这两个命令，他们的区别如下：

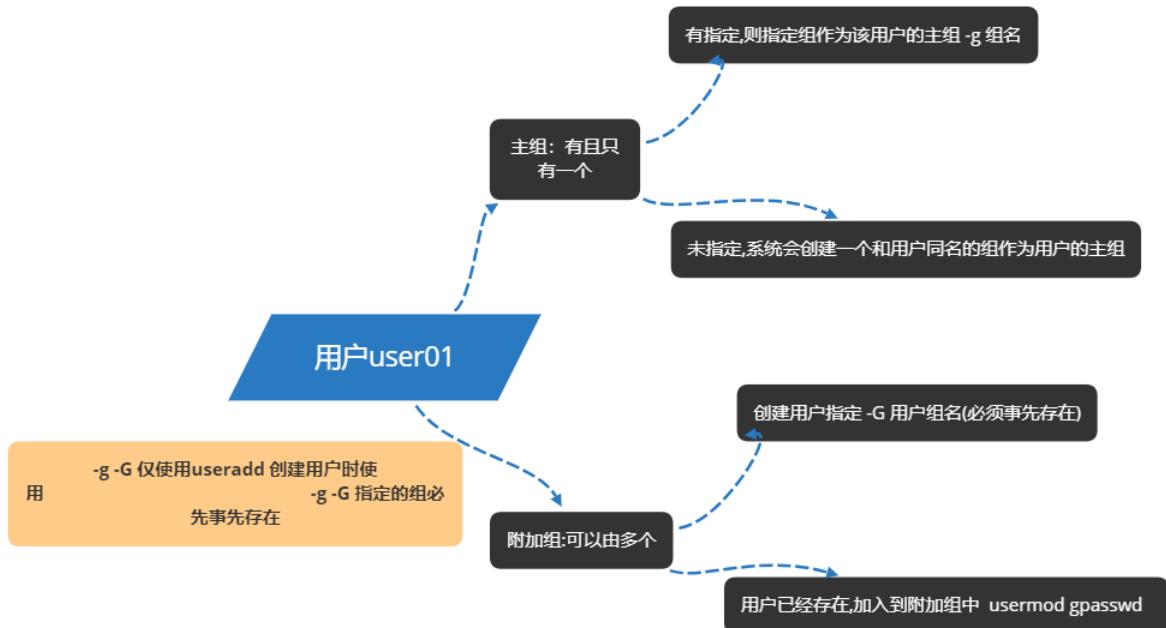
1. 使用useradd时，如果后面不添加任何参数选项，例如：#sudo useradd test创建出来的用户将是默认“三无”用户：一无Home Directory，二无密码，三无系统Shell。
2. 使用adduser时，创建用户的过程更像是一种人机对话，系统会提示你输入各种信息，然后会根据这些信息帮你创建新用户。

- adduser会提示设置密码，而useradd不会。
- adduser会创建用户目录，比如/home/freebird freebird是用户，useradd不会
- adduser会创建用户目录，比如/home/freebird freebird是用户，useradd不会
- adduser会询问全名，房间号码，电话号码等用户信息，useradd不会

usermod 可用来修改用户帐号的各项设定

- -c<备注> 修改用户帐号的备注文字。
- -d<登入目录> 修改用户登入时的目录。
- -e<有效期限> 修改帐号的有效期限。
- -f<缓冲天数> 修改在密码过期后多少天即关闭该帐号。
- -g<群组> 修改用户属组。
- -G<群组> 修改用户所属的附加组。
- -l<帐号名称> 修改用户帐号名称。
- -L 锁定用户密码，使密码无效。
- -s 修改用户登入后所使用的shell。
- -u 修改用户ID。
- -U 解除密码锁定。

shell是用户登录后运行的第一个程序。



userdel 删除用户 常用的选项是 **-r** (**--remove**) , 它的作用是把用户的主目录一起删除。

usermod 修改用户 **usermod -s /sbin/nologin** 用户 //修改指定用户的shell为nologin

gpasswd -a wo ziji 添加wo到ziji这个组中 添加什么用户到什么组

gpasswd -d wo ziji 将wo用户从ziji组中删除 删除什么用户从什么组

gpasswd -M wo,ni,ta ziji 将用户wo,ni,ta添加到ziji组中, **-M**批量添加用户到组

sudo 提权

将用户加入**wheel**组里, 普通用户才会有管理员权限

加入的方法 **useradd xxx -G wheel**

或**gpasswd -a xxx wheel**

创建的用户用管理员权限在使用前加**sudo** 例: **sudo useradd wo**

passwd 选项 用户名

-l 锁定口令, 即禁用账号。

-u 口令解锁。

-d 使账号无口令。

-f 强迫用户下次登录时修改口令。

Linux w命令用于显示目前登入系统的用户信息。

执行这项指令可得知目前登入系统的用户有哪些人, 以及他们正在执行的程序。

单独执行 **w** 指令会显示所有的用户, 您也可指定用户名称, 仅显示某位用户的相关信息。

语法

w [-fhlsuv] [用户名]

参数说明:

- f 开启或关闭显示用户从何处登入系统。
- h 不显示各栏位的标题信息列。
- l 使用详细格式列表，此为预设值。
- s 使用简洁格式列表，不显示用户登入时间，终端机阶段作业和程序所耗费的CPU时间。
- u 忽略执行程序的名称，以及该程序耗费CPU时间的信息。
- v 显示版本信息。

Linux 磁盘管理

Linux磁盘管理好坏直接关系到整个系统的性能问题。

Linux磁盘管理常用三个命令为df、du和fdisk。

- df: 列出文件系统的整体磁盘使用量 (df命令参数功能：检查文件系统的磁盘空间占用情况。可以利用该命令来**获取硬盘被占用了多少空间，目前还剩下多少空间等信息。**) 如果 df 没有加任何选项，那么默认会将系统内所有的(不含特殊内存内的文件系统与 swap)都以 1 Kbytes 的容量来列出来
 - -a : 列出所有的文件系统，包括系统特有的 /proc 等文件系统;
 - -k : 以 KBytes 的容量显示各文件系统;
 - -m : 以 MBytes 的容量显示各文件系统;
 - -h : 以人们较易阅读的 GBytes, MBytes, KBytes 等格式自行显示;
 - -H : 以 M=1000K 取代 M=1024K 的进位方式;
 - -T : 显示文件系统类型, 连同该 partition 的 filesystem 名称 (例如 ext3) 也列出;
 - -i : 不用硬盘容量，而以 inode 的数量来显示
- du: 检查磁盘空间使用量 (直接输入 du 没有加任何选项时，则 **du 会分析当前所在目录的文件与目录所占用的硬盘空间。**)
 - -a : 列出所有的文件与目录容量，因为默认仅统计目录底下的文件量而已。
 - -h : 以人们较易读的容量格式 (G/M) 显示;
 - -s : 列出总量而已，而不列出每个各别的目录占用容量;
 - -S : 不包括子目录下的总计，与 -s 有点差别。
 - -k : 以 KBytes 列出容量显示;
 - -m : 以 MBytes 列出容量显示;
- fdisk: 用于磁盘分区
 - -l : 输出后面接的装置所有的分区内容。若仅有 fdisk -l 时，则系统将会把整个系统内能够搜寻到的装置的分区均列出来。

磁盘格式化

mkfs [-t 文件系统格式] 装置文件名

-t : 可以接文件系统格式，例如 ext3, ext2, vfat 等(系统有支持才会生效)

磁盘检验

`fsck` (file system check) 用来检查和维护不一致的文件系统。

若系统掉电或磁盘发生问题，可利用`fsck`命令对文件系统进行检查。

```
fsck [-t 文件系统] [-ACay] 装置名称
```

`-t` : 给定档案系统的型式，若在 `/etc/fstab` 中已有定义或 `kernel` 本身已支援的则不需加上此参数

`-s` : 依序一个一个地执行 `fsck` 的指令来检查

`-A` : 对`/etc/fstab` 中所有列出来的 分区 (partition) 做检查

`-C` : 显示完整的检查进度

`-d` : 打印出 `e2fsck` 的 debug 结果

`-p` : 同时有 `-A` 条件时，同时有多个 `fsck` 的检查一起执行

`-R` : 同时有 `-A` 条件时，省略 / 不检查

`-V` : 详细显示模式

`-a` : 如果检查有错则自动修复

`-r` : 如果检查有错则由使用者回答是否修复

`-y` : 选项指定检测每个文件是自动输入yes，在不确定那些是不正常的时候，可以执行 `# fsck -y` 全部检查修复。

磁盘挂载与卸除

Linux 的磁盘挂载使用 `mount` 命令，卸载使用 `umount` 命令。

```
mount [-t 文件系统] [-L Label名] [-o 额外选项] [-n] 装置文件名 挂载点
```

```
umount [-fn] 装置文件名或挂载点
```

`-f` : 强制卸除！可用在类似网络文件系统 (NFS) 无法读取到的情况下；

`-n` : 不升级 `/etc/mtab` 情况下卸除。

vi / vim 键盘图



原图: www.viemu.com 翻译: fdl (linuxsir)

原图: www.viemu.com 翻译: fdl (linuxsir)

vim/vi 均在冒号模式中使用

vi 可以同时打开多个文件:
例: vim -o /etc/hosts /etc/hostname ^ww 切换文件(Ctrl+ww切换文件) (o垂直分割, w水平分割) :qall 全部退出

vimdiff 一个文件的路径 另一个文件的路径 比较两个文件的不同并会把不同用高亮显示

1. 移动光标

h|jk 向左右下上移动

[Ctrl] + [f] 屏幕『向下』移动一页, 相当于 [Page Down] 按键 (常用)

[Ctrl] + [b] 屏幕『向上』移动一页, 相当于 [Page Up] 按键 (常用)

[Ctrl] + [d] 屏幕『向下』移动半页

[Ctrl] + [u] 屏幕『向上』移动半页

+ 光标移动到非空格符的下一行

- 光标移动到非空格符的上一行

n<space> 那个 n 表示『数字』, 例如 20。按下数字后再按空格键, 光标会向右移动这一行的 n 个字符。例如 20<space> 则光标会向后面移动 20 个字符距离。

0 或功能键[Home] 这是数字『 0 』: 移动到这一行的最前面字符处 (常用)

\$ 或功能键[End] 移动到这一行的最后面字符处(常用)

H 光标移动到这个屏幕的最上方那一行的第一个字符

M 光标移动到这个屏幕的中央那一行的第一个字符

L 光标移动到这个屏幕的最下方那一行的第一个字符

G 移动到这个档案的最后一行(常用)

nG **n** 为数字。移动到这个档案的第 **n** 行。例如 **20G** 则会移动到这个档案的第 20 行(可配合 :set nu)

gg 移动到这个档案的第一行，相当于 **1G** 啊！(常用)

n<Enter> **n** 为数字。光标向下移动 **n** 行(常用)

2. 替换与搜索

/word 向光标之下寻找一个名称为 **word** 的字符串。例如要在档案内搜寻 **vbird** 这个字符串，就输入 **/vbird** 即可！(常用)

?word 向光标之上寻找一个字符串名称为 **word** 的字符串。

n 这个 **n** 是英文按键。代表重复前一个搜寻的动作。举例来说，如果刚刚我们执行 **/vbird** 去向下搜寻 **vbird** 这个字符串，则按下 **n** 后，会向下继续搜寻下一个名称为 **vbird** 的字符串。如果是执行 **?vbird** 的话，那么按下 **n** 则会向上继续搜寻名称为 **vbird** 的字符串！

N 这个 **N** 是英文按键。与 **n** 刚好相反，为『反向』进行前一个搜寻动作。例如 **/vbird** 后，按下 **N** 则表示『向上』搜寻 **vbird**。

使用 **/word** 配合 **n** 及 **N** 是非常有帮助的！可以让你重复的找到一些你搜寻的关键词！

:n1,n2s/word1/word2/g :范围 **s/old/new/选项(中间有无空格无所谓)**

例：**1,5 s/root/hao/** (等价于**1,5 s@root@hao@** 其他符号也可以 例 **!,#,,\$,%** 等 斜杠可以跟换成其他符号都无所谓) 在第一行到第五行中查找 **root** 并将它换成 **hao** **n1** 与 **n2** 为数字。在第 **n1** 与 **n2** 行之间寻找 **word1** 这个字符串，并将该字符串取代为 **word2**！举例来说，在 100 到 200 行之间搜寻 **vbird** 并取代为 **VBIRD** 则：『**:100,200s/vbird/VBIRD/g**』。(常用)

:1,\$s/word1/word2/g 或 **:%s/word1/word2/g** 从第一行到最后一行寻找 **word1** 字符串，并将该字符串取代为 **word2**！(常用)

\$在Linux里一般只最后的意思 %一般为全文的意思 **g**一般只全局 正则.*指的是整行 &引用查找的内容

:5,5 s/root/hao/ 从当前行到第五行查找 **root** 并更换为 **hao**

:4,9 s/^#/ 从第4行到第9行查找以#开头的并替换成 (空字符)

:5,10 s/.*/#/ 在第五行到第十行中找整行的并在行前面加个#

:5,10 s/.*/#/ 在第五行到第十行中找整行的并在行后面加个#

:1,\$s/word1/word2/gc 或 **:%s/word1/word2/gc** 从第一行到最后一行寻找 **word1** 字符串，并将该字符串取代为 **word2**！且在取代前显示提示字符给用户确认 (**confirm**) 是否需要取代！(常用)

查找到之后如果不想要高亮的话随便查找一个不存在的字符高亮就会消失！

3. 删除、复制与粘贴

Ctrl + N/P 自动补齐，选择的为灰色

x, X 光标当前所在的字符 在一行字当中，**x** 为向后删除一个字符 (相当于 **[del]** 按键)，**X** 为向前删除一个字符(相当于 **[backspace]** 亦即是退格键) (常用)

nx n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符，『10x』。

dd 删除 删除游标所在那一整行(常用)

nnd n 为数字。删除光标所在的向下 n 行，例如 20dd 则是删除 20 行 (常用)

d1G 删除光标所在到第一行的所有数据

dG 删除光标所在到最后一行的所有数据

d\$ 删除游标所在处，到该行的最后一个字符

d0 那个是数字的 0 ，删除游标所在处，到该行的最前面一个字符

D 从光标处删除到行尾

yy 复制 复制游标所在那一行(常用)

nny n 为数字。复制光标所在的向下 n 行，例如 20yy 则是复制 20 行(常用)

y1G 复制游标所在行到第一行的所有数据

yG 复制游标所在行到最后一行的所有数据

y0 复制光标所在的那个字符到该行行首的所有数据

y\$ 复制光标所在的那个字符到该行行尾的所有数据

p, P 粘贴 p 为将已复制的数据在光标下一行贴上，P 则为贴在游标上一行！ 举例来说，我目前光标在第

20 行，且已经复制了 10 行数据。则按下 p 后，那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？那么原本的第 20 行会被推到变成 30 行。 (常用)

J 将光标所在行与下一行的数据结合成同一行

c 重复删除多个数据，例如向下删除 10 行，[10cj]

u 撤销。(常用)

r 可以用来修改一个字符

R 连续替换

[Ctrl]+r 反撤销 redo(重做) 重做上一个动作。(常用)。

这个 u 与 [Ctrl]+r 是很常用的指令！一个是复原，另一个则是重做一次～ 利用这两个功能按键，你的编辑，嘿嘿！很快乐的啦！

. 不要怀疑！这就是小数点！意思是重复前一个动作的意思。 如果你想要重复删除、重复贴上等等动作，按下小数点『.』就好了！ (常用)

1. 指令行的储存、离开等指令

:w 存储到当前文件夹 将编辑的数据写入硬盘档案中(常用)

:w! 若文件属性为『只读』时，强制写入该档案。不过，到底能不能写入，还是跟你对该档案的档案权限有关啊！

:1,3 w /tmp/001.txt 将1到3行保存到/tmp/001.txt中

:w [filename] 例: :w /var/001.txt 另存为到 /var/001.txt 将编辑的数据储存成另一个档案 (类似另存新档)

:q 离开 vi (常用)

:q! 若曾修改过档案, 又不想储存, 使用 ! 为强制离开不储存档案。
注意一下啊, 那个惊叹号 (!) 在 vi 当中, 常常具有『强制』的意思~

:wq 储存后离开, 若为 :wq! 则为强制储存后离开 (常用)

ZZ 这是大写的 Z 哟! 如果修改过, 保存当前文件, 然后退出! 效果等同于(保存并退出)

ZQ 不保存, 强制退出。效果等同于 :q!。

:r [filename] :r /etc/hosts 将指定的文件内容读取到光标所放那个的位置 在编辑的数据中, 读入另一个档案的数据。亦即将 『filename』 这个档案内容加到游标所在行后面

:5 r /etc/hosts 将指定的文件内容读取到第五行之后

:n1,n2 w [filename] 将 n1 到 n2 的内容储存成 filename 这个档案。

:! command 暂时离开 vi 到指令行模式下执行 command 的显示结果! 例如『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息!

2.vim 环境的变更 (退出vi后再进入就没了)

:set nu或number 显示行号, 设定之后, 会在每一行的前缀显示该行的行号

:set nonu 与 set nu 相反, 为取消行号!

:set ic(ignore case的缩写 忽略大小写) 不区分大小写

:set noic 与ic相反

:set ai 自动缩进

:set list 显示控制字符 如: \$ 换行符 ^I Tab键 等

:set nolist 取消控制字符

永久配置vi环境 修改vi环境配置文件

/etc/vimrc 影响所有系统用户

~/.vimrc 影响某一个用户

命令模式:

- i,a,o,A切换到输入模式, 以输入字符。(o另起一行进入输入模式, a光标移到后一个进入输入模式, A 移到当前行的最后进去输入模式)
- x 删除当前光标所在处的字符。
- : 切换到底线命令模式, 以在最底一行输入命令。

输入模式:

- 字符按键以及Shift组合, 输入字符
- ENTER, 回车键, 换行

- **BACK SPACE**, 退格键, 删除光标前一个字符
- **DEL**, 删除键, 删除光标后一个字符
- **方向键**, 在文本中移动光标
- **HOME/END**, 移动光标到行首/行尾
- **Page Up/Page Down**, 上/下翻页
- **Insert**, 切换光标为输入/替换模式, 光标将变成竖线/下划线
- **ESC**, 退出输入模式, 切换到命令模式

冒号模式:

- q 退出程序
- w 保存文件

可视模式: (可进行可视化复制, 删除, 粘贴等 (y, d, p))

- v 可视模式
- V 可视行模式
- CTRL+v (^v) 可视块模式

sed非交互式文本编辑器, 常用于脚本。
vim为交互式文本编辑器。

Linux进程(内核调用进程)

ps静态的方式查看进程

top动态的实时的查看进程

进程标识:PID (process) , PPID(parent process ID父进程ID)

进程占用资源:disk IO (磁盘读写) , memory, CPU, network

STAT 进程状态:

R (running (正在运行) , runnable (等待运行)), T (暂停状态或跟踪状态,) , Z (Zombie 僵停) , X (dead死亡状态) , S (sleeping (可中断睡眠)) , D (不可中断的睡眠状态, 比如正在进行磁盘的IO操作) , K (KILLABLE) 。S,D,K都属sleeping状态, Z,X都属Zombie状态

Ss s进程的领导者, 父进程 S< <优先级较高的进程

SN N优先级较低的进程 R+ +表示是前台的进程组

SI 以线程的方式运行

例:用户调用程序文件的资源消耗 : 用户yu调用passwd 占用disk IO , memory, CPU, network(远程修改时)。生命周期:出生(父进程通过fork()函数产生的子进程), 执行(exec), 退出(exit)。

进程: 短进程:例: passwd、ls、cp、rm等运行时间短; 守护进程:例: sshd、httpd、vsftpd。

什么是进程?

进程是已启动的可执行程序的运行实例, 进程有以下组成部分:

已分配的内存的地址空间;

安全属性，包括所有凭证与特权；

程序代码的一个或多个执行线程；

进程状态。

程序:二进制文件，静态。/bin/date,/usr/sbin/httpd,/usr/sbin/sshd等

进程:是程序运行的过程，动态，有生命周期和运行状态。

第一个系统进程: centos5/6: init

centos7: systemd

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

在多任务处理操作系统中,每个CPU(或核心)在一个时间点上只能处理一个进程。在进程运行时,它对CPU时间和资源分配的要求会不断变化,从而为进程分配一个状态,它随着环境要求而变化。

```
[root@localhost ~]# ps aux
USER      PID %CPU %MEM      VSZ      RSS TTY      STAT START   TIME COMMAND
root         1  0.5  0.0    2900   1432 ?        Ss   Sep29   0:01 /sbin/init
USER运行进程的用户  PID进程ID  %CPU-CPU占用率  %MEM内存占用率  VSZ占用虚拟内存  RSS占用实际内存  TTY进程运行的终端  STAT进程状态  START进程的启动时间  TIME进程占用的CPU的总时间
COMMAND进程文件,进程名
vsz和rss的区别:以房子为例,总平方为120平方vsz,但实际可用只有100平方rss
```

ps:

```
ps aux --sort %cpu //以升序的方式显示cpu占用率排序
ps aux --sort -%cpu //以降序的方式显示cpu占用率排序
ps aux --sort %mem //以升序的方式显示mem占用率排序
ps aux --sort -%mem //以降序的方式显示mem占用率排序
ps aux --sort rss //以降序的方式显示rss排序
//按照上述方式皆可排序,若需降序在排序条件前加-即可
ps auxf 可以查看进程的父子关系

ps -ef |less
[root@localhost ~]# ps -ef |less
UID      PID  PPID  C STIME TTY          TIME CMD
root         1      0  0 Sep29 ?
00:00:01 /sbin/init

ps axo 字段 只显示指定的字段
例: 1.ps axo pid,ppid,user,%cpu //只显示字段pid,ppid,user,%cpu
2.ps axo pid,ppid,user,%cpu --sort -%cpu |less //只显示字段pid,ppid,user,%cpu,并以%cpu进行降序排序
3.ps axo pid,ppid,user,%cpu |grep httpd //只显示字段pid,ppid,user,%cpu,并过滤进程httpd显示

查看进程PID: 以sshd为例
1.ps aux |grep sshd
```

```
root      1963  0.0  0.0   8708  1036 ?          Ss   Sep29  0:00 /usr/sbin/sshd
2.pgrep -l sshd
1963 sshd
3.pgrep sshd
1963
4.pidof sshd
1963
```

top:

```
top -d(delay-time 延迟时间) 时间
例: top -d 1 //以1秒的时间刷新进程显示 默认为3秒刷新一次
top分两部分 上边部分为整体信息 下边的为进程信息
进入top中的命令:
h或? 帮助
M 按内存的使用排序
P 按CPU使用排序
N 以PID的大小排序
R 对排序进行反转显示
f 自定义显示字段
1 显示所有CPU的负载

w 保存top环境设置 //保存到当前用户家目录 ~/.toprc中
< 向前
> 向后

top -d 1 -p PID //查看指定进程的动态进程信息, 1秒刷新 p process
top -d 1 -u UID //查看指定用户的进程
top -d 1 -b -n 2 > top.txt(指定的文件) //将2次top信息写入指定文件中 -b -n 次数 查看多少次
top -d 1 -p 1326,1 //查看PID为1326和1的进程信息 1为systemd

PR() NI() VIRT(虚拟内存) RES(驻留内存, 程序代码本身使用的内存) SHR S %CPU(占用CPU)
%MEM(占用内存) TIME+(累计使用CPU, 单位为分秒) GROUP(组) COMMAND(程序文件)

load average: 2.92, 4.48, 5.20(加权值)
per-CPU load average :0.73 1.12 1.30 //以4核为例除以的负载状况
最近1分钟, 5分钟, 15分钟平均负载
例如现在CPU有4核, 那么2.92, 4.48, 5.20均除以4得到的就是当前CPU的平均负载情况, 若值大于1则代表处于过载状态, 若小于1则未负载 //一个cpu有4核则算4个cpu
```

信号控制进程

kill (PID or Job ID) killall (name)

pkill/pgrep [选项] pattern(模式) pkill可以给指定的进程发送信息, 它可以结束某个执行的进程或者目录登录的用户。

pgrep -u 用户(例: root,yu) //列出用户root和yu的所有进程

kill -l //可以查看当前系统所支持的全部信号

信号编号	信号名	信号简称 称	信号简介
1)	SIGHUP	HUP	重新加载配置(PID不变)(有些进程不支持重载)
2)	SIGINT	INT	键盘终端^C (CTRL+C)
3)	SIGQUIT	QUIT	键盘退出
9) (尽量少用)	SIGKILL	KILL	强制终止
15) 默认 kill 为 kill -15	SIGTERM	TERM	终止(正常结束),缺省信号, 允许自我清除
18)	SIGCONT	CONT	继续
19)	SIGSTOP	STOP	停止
20)	SIGTSTP	TSTP	暂停^Z

发送信号步骤,先查看PID, 然后用kill -编号(或信号名称或信号简称) PID

例如: 以1(HUP)为例: kill -1(-SIGHUP或-HUP) PID

```
[root@localhost ~]# ps aux |grep crond //查看进程号PID 2095
root      2095  0.0  0.0    5656  1300 ?          Ss   05:33   0:00 crond
[root@localhost ~]# kill -1 2095 //发送重新加载配置信号
[root@localhost ~]# ps aux |grep crond
root      2095  0.0  0.0    5656  1300 ?          Ss   05:33   0:00 crond //加载完
成后再次查询

[root@localhost ~]# kill -15 2095 //终止crond进程
[root@localhost ~]# ps aux |grep crond//查看进程 无
[root@localhost ~]# service crond start //开启crond进程
正在启动 crond:                                     [确定]
[root@localhost ~]# ps aux |grep crond //再次查看启动crond进程号 2961
root      2961  6.2  0.0    7276  1288 ?          Ss   06:01   0:00 crond

pkill -u yu //把yu用户的所有进程干掉
pkill -t TTY(终端)(tty1 为本地终端 pts/0 为伪终端(p pseudo 伪))(远程登录等) //仅
仅匹配终端, 以终端为参数杀掉匹配终端的所有进程 可用w命令查看登录状态 其中TTY为用户所登录的终
端
pkill -9 -t TTY(终端) //终止匹配终端上所有的进程, 并结束该终端

在top中输入k 可再输入信号控制编号同样可以结束进程
```

进程优先级 nice

Linux进程调度及多任务

相对优先级nice

查看进程的nice级别

启动具有不同nice级别的进程

更改现有进程的nice级别

Linux进程调度及多任务

每个CPU(或CPU核心)在一个时间点上只能处理一个进程,通过时间片技术, Linux实际能够运行的进程(和线程数)可以超出实际可用的CPU及核心数量。Linux内核进程调度程序将多个进程在CPU核心上快速切换,从而给用户多个进程在同时运行的印象。

相对优先级nice

由于不是每个进程都与其他进程同样重要,可告知进程调度程序为不同的进程使用不同的调度策略。常规系统上运行的大多数进程所使用的调度策略为SCHED_OTHER(也称为SCHED_NORMAL),但还有其它一些调度策略用于不同的目的。(sched调度)

SCHED_OTHER调度策略运行的进程的相对优先级称为进程的nice值,可以有40种不同级别的nice值。

nice值越高,表示优先级越低,比如+19,优先级低;

nice值越低,表示优先级越高,比如-20,该进程更不倾向于让出CPU。

查看进程的nice进程

1. 使用top查看nice级别

NI: 实际nice级别

PR: 将nice级别显示为映射到更大优先级队列, -20映射到0, +19映射到39.

2. 使用ps查看nice级别

例: ps axo pid,command,nice,cls --sort=-nice //按这些字段降序排序

TS表示该进程使用的调度策略为SCHED_OTHER. //cls字段会显示TS等调度策略

nice -n -5 sleep 6000& //用nice命令赋予创建的进程6000给-5的进程值,并在后台运行。&后台运行

更改现有进程的nice级别:

1. 使用top更改nice级别

r 调整进程的优先级(Nice Level)(-20高) --- 0 --- (19低)

//跟k的用法差不多,在top中按r输入PID然后输入进程值

2. 使用shell更改nice级别

[root@yu~]# sleep 7000 &

[3] 10089

[root@yu ~]# renice -20 10089 //renice 优先级 进程号 根据进程的进程号来改变进程的优先级

10089: old priority 0, new priority -20

进程作业控制 jobs shell默认只能运行一个命令

作业控制是一个命令行功能,允许一个shell实例来运行和管理多个命令。

如果没有作业控制,父进程fork()一个子进程后,将进入sleeping,直到子进程退出。

使用作业控制,可以选择性暂停,恢复,以及异步运行命令,让shell可以在子进程运行期间返回接受其它命令。

foreground(前台), background(后台),controlling terminal(控制终端)。

foreground: 前台进程是在终端中运行的命令,该终端为进程的控制终端。前台进程接收键盘产生的输入和信号,并允许从终端读取或写入到终端。

background: 后台进程没有控制终端,它不需要终端的交互。(不会接受任何输入输出操作)

```
[root@localhost ~]# sleep 5& //运行程序,让其在后台运行。
```

```
[1] 2837
```

```
[root@localhost ~]# sleep 10    sleep 秒数 //运行10秒
```

```
^Z                  //^Z,将前台程序挂起(暂停)到后台
```

```
[3]+  Stopped      sleep 10
```

```
jobs 查看后台运行的进程
1.fg %n 让后台运行的进程n到前台来
2.bg %n 让进程n到后台去; //PS:"n"为jobs查看到的进程编号.
[root@localhost ~]# jobs
[2]- Stopped sleep 5 [2][3]为作业号
[3]+ Stopped sleep 10

[root@localhost ~]# sleep 100 //运行一个进程
^Z //暂停该进程
[1]+ Stopped sleep 100
[root@localhost ~]# jobs //查看后台进程
[1]+ Stopped sleep 100
[root@localhost ~]# bg 1 //通过作业号, 让后台进程在后台运行
[1]+ sleep 100 &
[root@localhost ~]# jobs
[1]+ Running sleep 100 &

[root@localhost ~]# fg 1 //将后台进程调用到前台运行
sleep 100 //在前台运行期间占用终端, 父进程进入sleeping, 无法进行任何其他操作, 只能等待进程
结束 可按^C退出或^Z挂起
```

杀死后台程序方法:

```
[root@localhost ~]# ps aux |grep sleep //过滤查看sleep进程
root 2915 0.0 0.0 5676 496 pts/0 T 17:23 0:00 sleep 100
[root@localhost ~]# kill -9 2915 //查到后台程序pid 用kill杀死
[root@localhost ~]# ps aux |grep sleep
root 2950 0.0 0.0 6048 796 pts/0 S+ 17:28 0:00 grep sleep
[1]+ 已杀死 sleep 100
```

也可以用kill %工作号 杀死后台程序

```
[root@localhost ~]# jobs
[1]+ Stopped sleep 100
[root@localhost ~]# kill %1 只有kill后面可以跟%
[root@localhost ~]# jobs
[1]+ 已终止 sleep 100

kill %2 //杀死工作号为2的后台进程
kill 2 //杀死PID为2的进程
```

在vi/vim中若想干其它的事 按^Z 将vi/vim 挂起到后台

按fg 工作号 回到vi/vim中继续操作

默认只按fg会回到最近的那个被挂起到后台的程序 例如现在后台作业号有[1][2]两个后台程序, 值按fg会调用作业号为[2]的后台程序

```
[root@localhost ~]# vi wo
[1]+ Stopped vi wo
[root@localhost ~]# jobs //查看工作号
[1]+ Stopped vi wo
[root@localhost ~]# fg 1 //将工作号为1的进程调用到前台
vi wo
```

```
while :;do date;sleep 2;done //while循环每隔两秒钟刷新一次date
while :;do date;sleep 2;done & //后台的程序为done, 这是举例, 但其实因为do done 是一个整体, 所以这条命令也会位于后台进行, 但输出会显示在前台
因为;分号是用来阻隔排序命令的, 上面的操作&只会作用于done
(while :;do date;sleep 2;done) & //将前边的命令作为一个整体放在后台 程序放在了后台,
但输出仍在前台
fg 调到前台然后杀死
```

```
(while :;do date;sleep 2;done)&>/dev/null &//这个命令会将程序和程序的输出都放在后台运行，前台不会有显示。 第一个&不是后台符第二个才是
```

PS:一般使用;的长命令建议用()括起来当作整体使用
&放在命令的最后才是后台符

screen

Screen是一款由GNU计划开发的用于命令行终端切换的自由软件。用户可以通过该软件同时连接多个本地或远程的命令行会话，并在其间自由切换。提供了统一的管理多个会话的界面和相应的功能。

在Screen环境下，所有的会话都独立的运行，并拥有各自的编号、输入、输出和窗口缓存。用户可以通过快捷键在不同的窗口下切换，并可以自由的重定向各个窗口的输入和输出。

安装screen:

```
yum -y install screen
```

参数说明:

- A 将所有的视窗都调整为目前终端机的大小。
- d<作业名称> 将指定的screen作业离线。
- h<行数> 指定视窗的缓冲区行数。
- m 即使目前已在作业中的screen作业，仍强制建立新的screen作业。
- r<作业名称> 恢复离线的screen作业。
- R 先试图恢复离线的作业。若找不到离线的作业，即建立新的screen作业。
- s<shell> 指定建立新视窗时，所要执行的shell。
- S<作业名称> 指定screen作业的名称。
- v 显示版本信息。
- x 恢复之前离线的screen作业。
- ls或--list 显示目前所有的screen作业。
- wipe 检查目前所有的screen作业，并删除已经无法使用的screen作业。

常用screen参数

```
screen -S yourname -> 新建一个叫yourname的session(会话)
screen -ls          -> 列出当前所有的session
screen -r yourname -> 回到yourname这个session
screen -d yourname -> 远程detach某个session
screen -d -r yourname -> 结束当前session并回到yourname这个session
```

I/O重定向 (I/O Redirection)

重定向：重新改变文件的输出位置

stdin 标准输入 stdout标准输出 stderr错误输出

file descriptors (fd, 文件描述符 或Process I/O channels) (文件描述符为数字，如：0,1,2,3等)
\$\$当前进程的PID

文件描述符 0 代表标准输入(一般为键盘输入 键盘) 1代表标准输出(终端输出) 2代表标准错误或错误输出(终端输出) 描述符0只能从键盘只读，1, 2只能只写在终端 3+的为打开的文件，可读可写

进程关闭PID和文件描述符都会关闭，进程打开一个文件就会占用一个描述符，描述符用完了进程就不能继续打开文件除非把文件释放掉。

查看某个进程打开了哪些文件 ll /proc/PID/fd

ll /proc/\$\$/fd 查看当前bash打开了哪些文件

```
echo $$ 会输出当前bash的PID
```

```
date 1> date.txt 等同于 date > date.txt //date默认输出会显示在终端上，现在将date的输出重定向到date.txt文档中 //默认为文件描述符 1  
date 2> date.txt //因为文件描述符2为错误输出，但这里没有错误输出，所以这个命令的结果依旧会一描述符1标准输出的方式显示在终端屏幕上
```

```
ls /dadadadad //因为没有这个文件所以会显示错误  
ls /dadadadad 2>cuowu.txt //将错误的输出重定向到cuowu.txt中
```

```
ll -d /home > ll.txt //将home目录的信息的输出重定向到ll.txt中  
cat ll.txt //显示之前重定向的输出
```

```
< 输入重定向 >输出重定向
```

输入重定向：

```
正确输出： 1> 1>> 等价于 > >> 输出追加到文件内容的后边  
date > date.txt //将date的输出重定向到date.txt文件中并覆盖  
date >> date.txt //date的输出追加到date.txt内容的尾部  
[root@localhost ~]# vi ni  
[root@localhost ~]# cat < ni  
dadada  
[root@localhost ~]# cat > ni  
dada  
dad  
^C  
[root@localhost ~]# cat ni  
dada  
dad
```

```
ls /home /daadada >wo.txt 2>erro.txt //将正确的输出重定向到wo.txt，错误的输出重定向到erro.txt中 重定向的文件不用事先创建，重定向它会自动创建
```

将错误的输出和正确的输出重定向到同一个文件中：

```
ls /home/ /adadada &>wo.txt //& 混合输出 用 & 可以将错误与正确的输出重定向到同一文件中  
ls /home/ /adadada >wo.txt 2>&1 //先将文件描述符1重定向到wo.txt，再将2错误输出重定向到文件描述符1中，实现的效果跟上边一样，但原理不同，一般很少用
```

```
ls /home/ /adadadaa &>/dev/null 空设备，即将产生的输出丢掉  
ls /home/ /adadadaa >wo.txt 2>/dev/null //将正确的输出重定向到wo.txt中，错误的输出重定向到/dev/null中即丢掉。
```

脚本重定向演示：

```
ping c1 192.168.9.130 > /dev/null //将废话输出重定向到空设备里扔掉  
if [ $? -eq 0 ];then //c1 表示ping一次  
    echo "192.168.9.130 up" >>up.txt 将能ping通的放入up.txt  
else  
    echo "192.168.9.130 down" >>down 将ping不通的放入down.txt  
fi
```

若没有指定路径，默认会是当前目录下

要执行脚本前，要先赋予脚本一个执行权限

以ping1.sh为例：

```
chmod +x ping1.sh //赋予ping1.sh执行权限  
./ping1.sh //相对路径执行当前目录下的ping1.sh脚本
```

```
输入重定向：  
标准输入： < 等价于 0<  
mail -s "hello" yu < /etc/hostname //输入重定向，来自于文件，将文件的内容作为邮箱主题发送  
-s subject 主题 为hello yu为 要接受的用户  
mail yu < /etc/hostname 这样的格式发过去没有主题名
```

如果/dev/null设备被删除了恢复方法：（一般重启都会自动出来）

1.手动创建

mknod -m 666 /dev/null c 1 3 //mknod创建设备文件 权限666 c为字符设备 1为主设备号 3为从设备号 (b为有缓存的块设备, c, u为无缓存的块设备; p为管道设备) mknod同样也可以创建管道设备。

2.重启自动创建

关于邮件：

mail 查看邮件

mail 用户 发送某个用户发送邮件 例： mail yu //这个用户必须事先存在

Subject:XXX //邮件主题

XXX

XXX

XXX //邮件主体内容

. //退出邮件主体内容的编辑

EOT

用户收到邮件后查看方式：

先su到收邮件的用户 输入mail

邮件会有编号，输入相应的编号就会显示编号对应的邮件内容

mail d 邮件编号 //删除指定编号的邮件 例： d 1-30 删除邮件1-30

subshell(子shell)

命令在()中执行表示在子shell中执行

作用：如果不希望某些命令的执行对当前shell环境产生影响，请在subshell中执行。

```
[root@localhost ~]# cd /home;ls //在当前shell中执行  
wo yuchao  
[root@localhost home]# //shell变换为home目录了
```

```
[root@localhost ~]# (cd /home;ls) //在子shell中执行  
wo yuchao  
[root@localhost ~]# //当前shell还在家目录中
```

Linux yum 命令

yum (Yellow dog Updater, Modified) 是一个在 Fedora 和 RedHat 以及 SUSE 中的 Shell 前端软件包管理器。

基于 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载、安装。

进程管道 piping

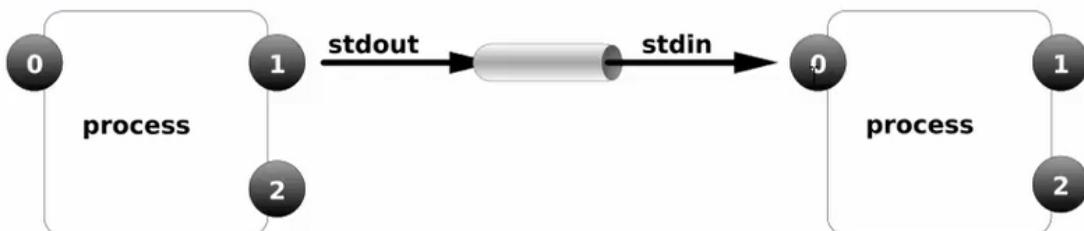
一般在命令中用的管道为匿名管道，只能在当前终端使用。

命名管道可以作用于其它终端。

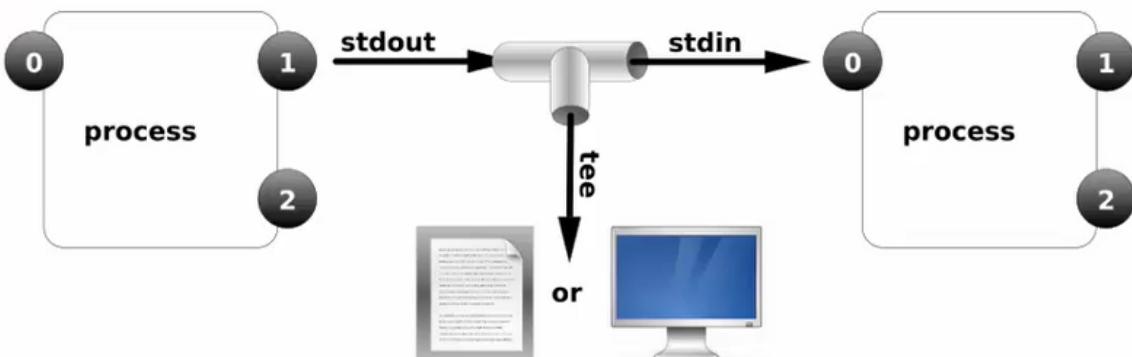
创建命名管道 mkfifo 名字。

重定向控制输出到文件中

管道控制输出(output)到程序(programs)中 也可以理解为重定向，只不过是将输出重定向到程序中进程输出通过管道送给下一个程序，作为下一个程序的输入。



tee管道



```
tee
-a 追加在文件内容的后边
若不加任何选项会覆盖内容
想保留哪段结果就在其后边加|tee -a 或 文件
df -P |grep '/' |awk '{print $5}' |tee df.txt |awk -F"%" '{print $1}' //tee将想
保留的东西重定向到一个文件中，这个案例中会将df -P |grep '/' |awk '{print $5}' 输出的结果
重定向到df.txt中
[root@localhost ~]# df -P |grep "/"
/dev/mapper/vg_yuchao-lv_root    17971068 5377116 12410068      31% /
[root@localhost ~]# df -P |grep "/"|awk '{print $5}'
31%
[root@localhost ~]# df -P |grep "/"|awk '{print $5}'|tee df.txt
31%
[root@localhost ~]# ls
```

```

anaconda-ks.cfg  df.txt  ffmpeg-3.4.1.tar.bz2  ping1.sh  post-install  post-
install.log
[root@localhost ~]# cat df.txt
31%  

date >date.txt //不会在屏幕显示结果
date |tee date.txt //会在屏幕显示结果，同样也会将结果保存到date.txt中

```

sort命令：

```

[root@yu ~]# sort -t":" -k3 -n /etc/passwd      //以:分隔，将第三列按数字升序。
[root@yu ~]# sort -t"." -k3 -n /etc/passwd -r    //逆序 以.为分隔符，第三列按数值进行
逆序排列 -r 逆序
[root@yu ~]# sort -t":" -k3 -n /etc/passwd | head //以:为分隔符，第三列按数值进行升序
排列，并打印出前10行，head默认打印前10行。
-t指定字段分隔符 -field-separator
-k指定列
-n按数值
-v 取反
-r 逆序

```

举例：

打印最占CPU的前5个程序：

```

ps aux --sort=-%cpu |head -6 // -6是因为第一行为描述文字
[root@localhost ~]# ps aux --sort=-%cpu |head -6
USER      PID %CPU %MEM     VSZ   RSS TTY      STAT START   TIME COMMAND
root      2646  8.0  0.0   5052  1024 pts/1    R+   01:26   0:00 ps aux --sort=-
%cpu
root       1  0.0  0.0   2900  1428 ?        Ss   00:16   0:01 /sbin/init
root       2  0.0  0.0     0    0 ?        S    00:16   0:00 [kthreadd]
root       3  0.0  0.0     0    0 ?        S    00:16   0:00 [migration/0]
root       4  0.0  0.0     0    0 ?        S    00:16   0:00 [ksoftirqd/0]
若不想要第一行描述，可以将其过滤掉：
ps aux --sort=-%cpu |head -5 |grep -v '%CPU'

```

统计当前/etc/passwd中用户使用的shell类型

思路：取出第七列(shell)|排序(把相同归类)|去重。 //uniq 相邻去重 相邻的多个1保留一个1 -c 去重的同时显示有多少个相同的1

```

[root@tianyun ~]# awk -F: '{print $7}' /etc/passwd
[root@tianyun ~]#
awk -F: '{print $7}' /etc/passwd |sort
[root@tianyun ~]# awk -F: '{print $7}' /etc/passwd |sort |uniq
[root@tianyun ~]# awk -F: '{print $7}' /etc/passwd |sort |uniq -c
F: 指定字段分隔符
$7 第七个字段

```

awk默认是以空格或tab为分割。

统计网站的访问情况top 20:

思路：打印所有访问的连接|过滤访问网站的连接|打印用户的IP|排序|去重

```

[root@tianyun ~]# ss -an |grep :80 |awk -F ":" '{print $8}' |sort |uniq -c
[root@tianyun ~]# ss -an |grep :80 |awk -F ":" '{print $8}' |sort |uniq -c |sort
-k1 -rn |head -n 20

```

打印当前所有ip:

```
ip addr |grep 'inet ' |awk '{print $2}' |awk -F"/" '{print $1}'  
ip a=ip addr=ip address  
//过滤'inet '， 打印第二列，以/为分割打印第一列  
  
打印根分区已用空间的百分比（仅打印数字）：  
df -P |grep '/$' |awk '{print $5}' |awk -F"%\"'{print $1}'    过滤以/结尾的，再打印第五列，再按%分割，再打印第一列
```

yum 语法

```
yum [options] [command] [package ...]
```

options: 可选，选项包括-h（帮助），-y（当安装过程提示选择全部为 "yes"），-q（不显示安装的过程）等等。

command: 要进行的操作。

package: 安装的包名。

yum常用命令

1. 列出所有可更新的软件清单命令: `yum check-update`
2. 更新所有软件命令: `yum update`
3. 仅安装指定的软件命令: `yum install <package_name>`
4. 仅更新指定的软件命令: `yum update <package_name>`
5. 列出所有可安装的软件清单命令: `yum list ***` (找出包含***开头的软件名称有哪些)
6. 删除软件包命令: `yum remove <package_name>`
7. 查找软件包命令: `yum search <keyword>`
8. 清除缓存命令:

`yum clean packages`: 清除缓存目录下的软件包

`yum clean headers`: 清除缓存目录下的 headers

`yum clean oldheaders`: 清除缓存目录下旧的 headers

`yum clean, yum clean all (= yum clean packages; yum clean oldheaders)` :清除缓存目录下的软件包及旧的 headers

Linux apt 命令

apt (Advanced Packaging Tool) 是一个在 Debian 和 Ubuntu 中的 Shell 前端软件包管理器。

apt 命令提供了查找、安装、升级、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。

apt 命令执行需要超级管理员权限(root)。

apt 语法

```
apt [options] [command] [package ...]
```

options: 可选，选项包括 `-h`（帮助），`-y`（当安装过程提示选择全部为"yes"），`-q`（不显示安装的过程）等等。

command: 要进行的操作。

package: 安装的包名。

apt 常用命令

列出所有可更新的软件清单命令: `sudo apt update`

升级软件包: `sudo apt upgrade`

列出可更新的软件包及版本信息: `apt list --upgradeable`

升级软件包, 升级前先删除需要更新软件包: `sudo apt full-upgrade`

安装指定的软件命令: `sudo apt install <package_name>`

安装多个软件包: `sudo apt install <package_1> <package_2> <package_3>`

更新指定的软件命令: `sudo apt update <package_name>`

显示软件包具体信息, 例如: 版本号, 安装大小, 依赖关系等等: `sudo apt show <package_name>`

删除软件包命令: `sudo apt remove <package_name>`

清理不再使用的依赖和库文件: `sudo apt autoremove`

移除软件包及配置文件: `sudo apt purge <package_name>`

查找软件包命令: `sudo apt search <keyword>`

列出所有已安装的包: `apt list --installed`

列出所有已安装的包的版本信息: `apt list --all-versions`

组合命令 ***&&***

补充:

因为在脚本中不能手动操作^D结束, 所以一般会创建一个结束标志, 一般为EOF, 也可以自己创建。<<EOF
因为一般为了美观会有缩进, 而有了缩进 EOF会识别不出来, 所以为了能够正确结束需要 <<-EOF 在
创建结束符之前价格- 表示有Tab缩进。 缩进用Tab缩进。

linux 一切皆文件 {参数1, 参数2, ...}大括号可以实现批处理

* 通配符(不包含隐藏文件) 比如: /home下有wo, wo1, wo2 rm -r wo* 删除以wo开头的所有文
件

/home下有001.txt, 002.txt, 003.txt rm -r *.txt 删除以.txt结尾的所有文件

shell是系统的用户界面, 提供了用户和内核进行交互操作的接口, 本质是命令解释器。

overwirte 覆盖 **release** 发行 一般网站找帮助 [documentation/docs](#) 若没有在页面找到有可能在**download**里

/etc/bashrc shell的配置文件之一
分页显示 如: `date --help |less`

ctrl+shift+t 新增一个终端

ctrl+l (clear) 清屏

ctrl+d 退出 等价于**exit**

ctrl+c 终止当前运行到程序

ctrl+a 光标移动到命令行的最前端

ctrl+e 光标移动到命令行的后端

ctrl+u 删除光标前所有的字符

ctrl+k 删除光标后所有的字符

ctrl+r 搜索历史命令, 利用关键字

Alt+. . 引用上一个命令的最后一个参数, 等价于!\$

Esc . 先按esc然后松手按., 引用上一个命令的最后一个参数, 等价于!\$

echo \$[65954480/1024/1024] 计算值的方法 **echo \$[]**

历史命令 **history !** 表示选择 ! 22 表示选择历史第22条命令

history -c //清空历史命令

! 字符串 搜索的历史命令必须是以xxx开头到命令 键盘上下键也可翻历史命令

任何表达式前都可加!来表示取反。例: **find /etc ! -name "wo.txt"** //在etc下找文件名不是wo.txt的文件或目录

凡是以前缀点开头到文件都是隐藏文件 以点开头的文件名, 点也是其一部份 例: .ch 文件名就是 .ch

命令选项一般以 -/--开头 参数为命令作用到对象

给命令起别名

alias 查看系统的别名

alias yu='ls -a /home' //yu代替了ls -a /home (临时的, 仅在当前shell生效)

unalias取消别名 **unalias yu** //取消上边创建到yu别名

查看命令是不是别名 **type -a 命令** 比如: **type -a -l**

若不想用别名在命令前加\ /表示跳过 比如: \ls 就只会以黑颜色显示文件及文件夹

别名优先于选项 比如: cp一般代表的是 cp -i 提醒用户是否覆盖 cp -f 强制覆盖不提醒, 仍旧会提醒是否覆盖, 因为cp 代表的是cp -i别名优先于cp -f选项

永久别名 **gedit /etc/bashrc** 打开一个类是记事本的文件, 拉到最后用 alias (例: alias yu='ls -a /home') 取自己的别名 保存, 退出, 重新打开终端, 取的别名就可以用了, 属于永久别名。

linux获得帮助 在帮助里 [] 为可选项 | 或者(多选一)

1.命令 **--help**

2.**man (manual 手册) 手册 man man** 查看man如何使用 命令帮助: 章节1, 8 函数帮助: 章节2, 3 文件格式帮助: 章节5 man共有10章

man -f passwd 列出所有章节中的passwd手册 **man 1 passwd** passwd命令的帮助 **man 5 passwd** 用户配置文件的帮助

man里的**EXAMPLES** 是讲述命令的用法的。

redhat官方帮助: <http://docs.redhat.com> linux的系统帮助指南

mysql官方 : <http://www.mysql.com> mysql帮助

用时间创建文件的名字

比如: **touch `date +%F`_file.txt** 比如当前时间为2020.9.20那么创建的文件名为:
2020.9.20_file.txt

两种时间： 系统时间：即系统时间 硬件时间：即主板**bios**时间

主机名在 /etc/hostname

生成密钥给服务器，以后登录不用密码也可以：

```
ssh-keygen //生产密钥  
ssh-copy-id 服务器ip地址 //将本地密钥拷贝到服务器
```

which 查看命令位于哪里 **whoami** 查看当前用户 我是谁

tree 命令 树 **tree -d 路径** 只看文件夹不看文件 **tree -L 2 路径** 只看两层

wc命令可以查看文件有多少行

wc 文件/文件路径 wc -l 只查看行数

-c或--bytes或--chars 只显示Bytes数

-l或--lines 只查看行数

-w或--words 只查看字数

--help 帮助

--version 显示版本信息

```
yum -y install epel-release
```

```
yum -y install sl
```

sl 可以装个小火车

at命令：设置计划任务

atq 查看所有计划任务列表(里面的编号可以用来关闭计划任务)

atrm 删除计划任务 **atrm 编号** //删除编号为xxx的计划任务

at 例：**at now+1min** //计划任务，当前时间加1分钟，到时之后运行计划的任务 //**at> at**计划任务的提示符

```
at>touch file //创建文件
```

```
at 11:01 //计划在11:01的时候执行计划任务
```

```
at>touch file //要执行的计划任务
```

设置好计划任务后按**Ctrl+D**退出

要**base**补全功能需要装一个**base-com***的包： **yum -y install base-com***

若想知道命令由哪些单词组成的可以用**man**命令查看 例：**man acl**

查看系统使用信息：**uptime** //登录时间，登录用户等

查看**SSH**是否安装（检查是否装了**SSH**包）。

输入命令：**rpm -qa | grep ssh** 或者 (**rpm -qa | grep openssh**)

查看**SSH**服务是否正在运行。

输入命令：**/etc/init.d/sshd status** 或者 (**service sshd status**)

netstat -antp | grep sshd 查看是否启动22端口

开启**ssh**服务：**service sshd start**

关闭**ssh**服务：**service sshd stop**

重启**ssh**服务：**service sshd restart**

安装**ssh**：**yum install openssh-server**

CentOS7 重启ssh服务的命令如下:

```
# service sshd restart  
出现:Redirecting to /bin/systemctl restart sshd.service  
需要启动该服务:  
systemctl start sshd.service  
重启 sshd 服务:  
systemctl restart sshd.service  
设置服务开启自启:  
systemctl enable sshd.service
```

centos6的操作:

```
关闭防火墙: service iptables stop  
开启防火墙: service iptables start  
重启防火墙: service iptables restart  
查看防火墙状态: service iptables status  
  
永久关闭防火墙: chkconfig iptables off  
永久开启防火墙: chkconfig iptables on  
查看状态: chkconfig --list iptables
```

磁盘:

SSD 固态硬盘

HDD 机械硬盘

机械硬盘:

1.扇区(sector): 扇形的区域, 以前为512byte, 现在为4K.

2.柱面(cylinder): 多张盘片, 半径相同的磁道组成的柱体称为柱面.

3.磁道(track):读写硬盘时, 磁头不动, 磁盘是旋转的, 则连续写入的数据是排列在一个圆周上的。我们称这样的圆周为一个磁道(Track)。

磁头不动, 就是在一个磁道 上读写,磁头移动, 就会在不同磁道上读写

根据硬盘规格的不同, 磁道数可以从几百到数千不等,一个磁道上可以容纳数KB的数据(一个track包含数个sector)

现在分区以扇区分区, 在以前以柱面分区。

固态硬盘:

第一, SSD不需要机械结构,完全的半导体化,不存在数据查找时间、延迟时间和磁盘寻道时间,数据存取速度快。

第二, SSD全部采用闪存芯片,经久耐用,防震抗摔,即使发生与硬物碰撞,数据丢失的可能性也能够降到最小。

第三,得益于无机械部件及FLASH闪存芯片, SSD没有任何噪音,功耗低。

第四,质量轻,比常规1.8英寸硬盘重量轻20-30克,使得便携设备搭载多块SSD成为可能。同时因其完全半导体化,

无结构限制,可根据实际情况设计成各种不同接口、形状的特殊电子硬盘。

硬盘尺寸: 3.5, 2.5, 1.8 //台式机一般为3.5英寸。

插拔方式: 热插拔, 非热插拔。

硬盘接口：IDE硬盘已被淘汰

SATA(Serial ATA)

SAS(Serial Attached SCSI)即串行连接SCSI

PCIE(), FC(光纤通道/接口)

SATA接口在两个金手指之间有个缺口，SAS接口两个金手指之间是实心的。

硬盘设备命名

物理硬盘 : /dev/sd[a-z]

KVM虚拟化 : /dev/vd[a-z] (半虚拟化驱动)
/dev/sd[a-z] (全虚拟化驱动) |

KVM增加硬盘

半虚拟化驱动磁盘 : online

全虚拟化驱动磁盘 : offline

HP服务器硬盘

/dev/cciss/c0d0

/dev/cciss/c0d0p1 //c0第一个控制器, d0第一块磁盘, p1分区1

/dev/cciss/c0d0p2 //c0第一个控制器, d0第一块磁盘, p2分区2

存储连接方式：

本地存储

外部存储: scsi线, sata线, sas线, FC线

网络存储: 以太网(iscsi,glusterFS,ceph(分布式存储)), FC网络 //集中式存储和分布式存储

分区方式：

MBR分区表 <2TB fdisk 4个分区(4个主分区, 扩展分区, 逻辑分区) 最多4个主分区 例: 3主+1扩展(n逻辑)

GPT分区表 大于2TB均可选 gdisk(parted centos6之前用的GPT分区软件) 128个主分区

MBR与GPT互相转换会导致数据丢失。

基本分区管理：(基本分区不能扩容)

lsblk 查看硬盘

ll /dev/vd* 或sd* //*[vs] 可以是vd* 也可以是sd*

echo \$[扇区数/2/1024/1024] //一个扇区为512bytes 2*扇区=1KB 扇区数/2=KB /1024=MB
/1024=GB

创建GPT分区：

若没gdisk需要先下载, yum -y install gdisk

gdisk /dev/要分区的磁盘 -l 查看分区信息

创建MBR分区：

fdisk -l 查看磁盘信息

fdisk /dev/分区名 //为分区操作

partprobe /dev/要更新的分区 //手动让内核在不需要重启的情况下，更新分区表，为新设备创建设备文件。

一个主分区，剩下的扩展分区都给，用于创建逻辑分区。

MBR 的Disk label type:dos;

GPT 的Disk label type:gpt.

创建文件系统(格式化) centos7默认为xfs:

mkfs.xfs /dev/xxx //xfs的格式化和创建很快 //L 可以指定卷标 tune2fs -L tune2fs -L XXX /dev/XXX 可以修改卷标

mkfs.ext4 /dev/xxx 或mkfs -t ext4 /dev/xxx //ext为扩展的意思

挂载：

mount 挂载 -t 表示挂载的设备是什么类型或什么文件系统(xfs,ext4等) -o 表示什么选项 -o ro 只读 //临时的挂载 不指定类型为auto, 不指定选项为defaults

文件系统类型：vfat, ext4, xfs, (网络文件系统：)nfs,cifs,(镜像:)iso9660,本地回环loop。

umount 取消挂载 后可加设备也可加挂载的目录

自动挂载：可以用UUID或设备名挂载 推荐用uuid挂载，因为设备名会变但uuid不会改变

/etc/fstab 自动挂载的目录

每个分区都有一个UUID 可以用blkid查询

blkid 获得当前所有设备的UUID blkid /dev/设备名 获得指定设备的UUID

挂载iso镜像文件：

dd if=/dev/cdrom of=/XXX.iso 或 dd < /dev/cdrom > /XXX.iso //将光盘做成iso镜像 做前先将光盘放在光驱中。

自己制作iso，例把/etc做成etc.iso:

genisoimage -o /tmp/etc.iso -r /etc //将/etc输出到/tmp/etc.iso

file /tmp/etc.iso //查看文件类型

挂载：mount /tmp/etc.iso /挂载目录/ //光盘目录为只读

挂载：1.可以使用UUID挂载

2.可以使用设备名挂载 (逻辑卷使用设备名挂载就好)

3.可以使用卷标挂载 LABEL=XXX

4.可以使用UDEV 可以给设备起一个别名 aclice

```

vim /etc/fstab
UUID=a135856a-aaa1-40b1-96cd-e77e01ccf728 /
xfs
defaults 0 0
UUID/设备名 挂载点 文件系统类型(xfs等) defaults(默认为rw、ro、rw)挂载选项 0
不备份 0不检测

default默认的值为rw, uid, dev, exec, auto, nouser, async(异步)。
rw 读写 ro 只读 uid 支持uid dev 支持设备文件
nodev 不支持设备文件 noexec 不允许执行二进制文件 exec 允许执行二进制文件 auto mount
-a 开机自动挂载 nomount mount -a开机不自动挂载 async 异步写入 sync 同步写入
usrquota 支持用户级磁盘配额功能 grpquota 支持组级磁盘配额功能 acl 支持acl功能
(centos7 默认就有, 6无) remount 在线重新加载 pir 指定优先级

mount -o rw,remount /xxx //挂载目录, 在线重新加载挂载的目录为读写

查看某个设备的具体挂载信息:
mount |grep 设备名

```

mount -a 会读取fstab文件，它能挂上说明开机也能挂上。

mount 直接回车可以查看文件系统的权限(只读, 读写等)

//块 为文件存储的最小单元

//挂载目录必先事先准备。挂载就是把一个分区和目录相关联。挂载之前要先格式化。

挂载：Linux 系统中“一切皆文件”，所有文件都放置在以根目录为树根的树形目录结构中。在 Linux 看来，任何硬件设备也都是文件，它们各有一套自己的文件系统（文件目录结构）。因此产生的问题是，当在 Linux 系统中使用这些硬件设备时，只有将 Linux 本身的文件目录与硬件设备的文件目录合二为一，硬件设备才能为我们所用。合二为一的过程称为“挂载”。

如果不挂载，通过Linux系统中的图形界面系统可以查看找到硬件设备，但命令行方式无法找到。

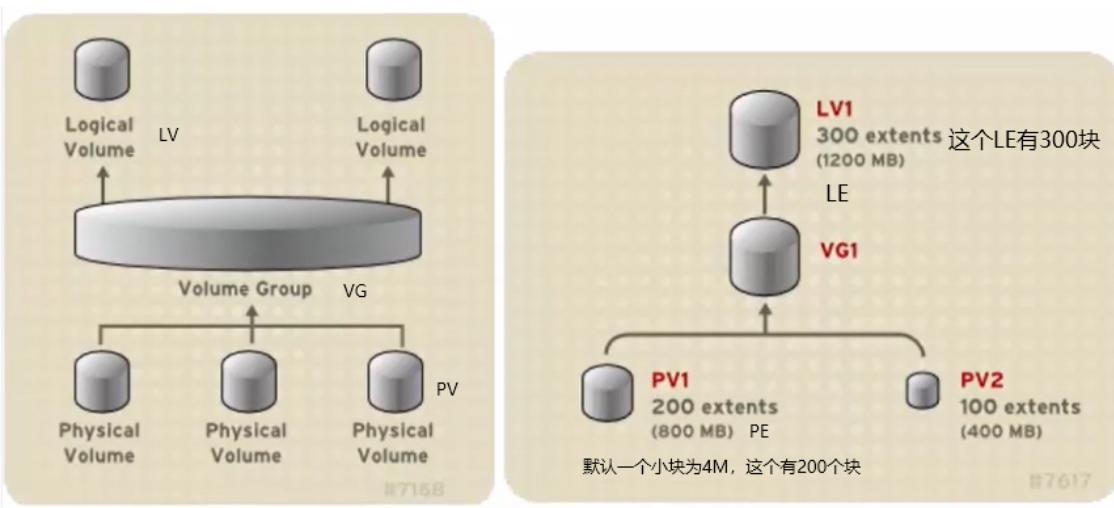
挂载，指的就是将设备文件中的顶级目录连接到 Linux 根目录下的某一目录（最好是空目录），访问此目录就等同于访问设备文件。

由于挂载操作会使得原有目录中文件被隐藏，因此根目录以及系统原有目录都不要作为挂载点，会造成系统异常甚至崩溃，挂载点最好是新建的空目录。

若根目录存储空间不足，但还想在某个目录里多存点数据，可以用挂载：

1. 创建一个新分区；2.挂载新分区 到 临时位置 例：/disk1；3.cp -rf /要存数据的目录/* /disk1 ；4.umount /disk1 ；5./etc/fstab 新分区

LVM逻辑卷管理(LV 逻辑卷 logical volume)



PV (physical volume 物理卷) VG (volume group 卷组) PE (physical extent 物理扩展) : 默认为4MB的小块，也可以为8M,16M等。创建VG时指定，-s可以更改指定的小块大小如：-s 8MB. LE (logical extent 逻辑扩展)

逻辑扩展映射在物理扩展上。逻辑卷由很多LE组成的。pe针对卷组，le针对逻辑卷。 le, pe是扩展的最小单元

1.lvm可以在线进行扩容 (online) 。

2.数据迁移，逻辑卷可以将多个物理磁盘合成一个分区。 (online)

一般做逻辑卷前先做RAID，逻辑卷没有容错，

查看pv, lv, vg信息: pv/lv/vgs, pv/lv/vgdisplay, pv/lv/vgscan s简化版信息, scan稍微长点的信息, display详细信息。

1.PV, VG,LV

创建PV : pvcreate /dev/vda //以vda为例

vgcreate datavg /dev/vda //将vda加入卷组datavg中

lvcreate -L 100M -n lv1 datavg // -L 指定lv的大小为100M, -n 指定lv的名字为lv1 datavg 从哪个卷组中选择

lvcreate -l 100%FREE -n lv1 datavg //从datavg卷组中创建lv1，并将VG空余的所有空间分给lv1 //FREE剩余空间 也可以 50%FREE或50%VG 等

lvcreate -l 25 -n lv1 datavg // -l 指定lv的大小为25个le, 一个le默认大小为4M

这时要查看lv的路径为 lvdisplay /dev/datavg/lv1

格式化挂载：逻辑卷可以不用uuid挂载

格式化 mkfs.xfs /dev/datavg/lv1

挂载：mkdir /mnt/lv1 //创建一个要挂载的空目录

vim /etc/fstab //进入etc/fstab 永久挂载

/dev/datavg/lv1 /mnt/lv1 ext4 defaults 0 0

mount -a //查看是否挂载成功

2.VG扩容和缩小 扩容 vgextend 缩小 vgreduce

VG扩大的实际意义是为了扩大LV，减小是为了收缩空间。

扩容前先查看lv属于哪个VG lvs

再查看vg容量还有多少 vgs /dev/vdb 为例子

先把/dev/vdb变为pv，再把它通过vgextend 扩大哪个vg 再将扩容的材料给它 例：现在有/dev/vdb pvcreate /dev/vdb -> vgextend datavg /dev/vdb 或者 vgextend datavg /dev/vdb 系统会自动先将/dev/vdb变为pv

缩小vg先要做数据的迁移，不能将有数据的PV从VG中删除。

pvmov 移动物理扩展，数据迁移 例：要把vda的数据交接到vdb

pvmov /dev/vda /dev/vdb //若不写迁移对象会默认迁给下一个

vgreduce 减小vg vgreduce datavg /dev/vda //移除vda,减小了vg的容量。

删除lv: lvremove /dev/卷组/lv名 -f强制删除

3.LV扩容及缩小 缩小不建议 //扩容时最低为4M，低于4M会按4M处理

lvextend -L 800M /dev/datavg/lv1 //总容量到800M

lvextend -L +800M /dev/datavg/lv1 //在原有容量上增加800M

lvextend -l 15 /dev/datavg/lv1 //最终到15个le

lvextend -l +15 /dev/datavg/lv1 //增加15个le

FS扩容（文件系统扩容 file system）：

FS扩容需先看是什么文件系统(ext4,xfs)

扩文件系统xfs使用的命令：xfs_growfs /dev/datavg/lv1 大小 //若不加大小会自动填满，一般填满就行

扩文件系统ext2/3/4使用的命令:resizefs /dev/datavg/lv1 大小 //若不加大小会自动填满

扩逻辑卷分两步：1.扩逻辑卷；2.扩文件系统

扩容详细查看：<https://blog.csdn.net/ninghao2015/article/details/73382405>

LVM 快照snapshot(快照卷与原始卷必须在同一VG中) (快照拷贝的为元数据)

lvcreate -L 50M -n lv1-snap -s /dev/datavg/lv1 //ext4

lv1的快照卷 lv1-snap 自己命名的快照卷名，-s 创建快照的选项，之后跟原始卷的路径 原卷(Original) 快照卷(Snapshot)

lvcreate -o nouuid /dev/datavg/lv2-snap /mnt/lv2-snap //xfs挂载快照需要加-o nouuid 因为xfs不支持uuid

交换分区管理 Swap (交换分区可以为分区，逻辑卷，文件)

提升内存容量，防止OOM (out of memory)

查看交换分区 free -m free不加m选项，显示的为KB，加-g显示GB，加m为MB

swapon -s //查看交换分区

top -d 1

增加交换分区：准备分区如fdisk /dev/XXX (按t 转换分区 82 也可以不转换)

格式化分区mkswap /dev/XXX

挂载：1.先获得其uuid， blkid /dev/XXX

2.vim /etc/fstab

uuid="" swap swap defaults 0 0

swapon -a //激活所有交换分区(读取/etc/fstab)

swapoff /dev/XXX //关闭某个交换分区

```
swapon /dev/XXX //激活某个交换分区
```

逻辑卷做交换分区：

```
lvcreate -L 200M -n lv-swap datavg //创建一个逻辑卷
```

```
mkswap /dev/datavg/lv-swap //格式化逻辑卷为交换分区
```

逻辑卷没必要使用UUID，直接用路径就行/dev/datavg/lv-swap到/etc/fstab里添加
/dev/datavg/lv-swap swap swap defaults 0 0

```
swapon -a swapon -s
```

文件做交换分区：

```
dd if=/dev/zero of=/swap.txt bs=1M count=512
```

//dd：用指定大小的块拷贝一个文件，并在拷贝的同时进行指定的转换

if=文件名：输入文件名，缺省为标准输入。即指定源文件。of=文件名：输出文件名，缺省为标准输出。即指定目的文件。bs=bytes：同时设置读入/输出的块大小为bytes个字节。

count=blocks：仅拷贝blocks个块，块大小等于ibs指定的字节数。512M

```
ll /swap.txt ll /swap.txt -h //查看文件信息
```

```
mkswap /swap.txt //格式化文件为交换分区
```

复制路径到 /etc/fstab //vim /etc/fstab

```
/swap.txt swap swap defaults 0 0
```

```
swapon -a //如果出现问题，修改文件权限就行 chmod 权限值 /swap.txt
```

//交换分区一般设置好，要用的话会先用第一个再用第二个，以此来使用。若需要同时使用可以在defaults后加,pri=1,相同的优先级(数字必须相同)就行，例：

```
uuid="" swap swap defaults,pri=1 0 0
```

```
/swap.txt swap swap defaults,pri=1 0 0
```

//没有文件系统的分区为裸设备。格式化分区就是为了创建文件系统。

文件的所有信息都存储在inode中，数据存储在data blocks中。查看inode的编号 ll -i 或 df -i
inode表被占光就不能创建文件了。

inode:记录文件的属性，(文件的元数据 metadata)，一个文件占用一个inode，同时记录此文件数据所在的block number。

inode table: 存储文件的元数据。如：文件的权限(r,w,x),文件的属主/属组(owner/group),文件的大小，文件的改变时间(ctime)，修改时间(mtime)，访问时间(atime)，文件所在的block number。

superblock:记录此文件系统的整体信息，包括inode/block的总量、使用量、剩余量，以及文件系统的格式等。没有superblock文件系统就不能正常使用，可以用后边的超块来恢复。每个block group 都可能含有superblock。

block: 实际存储的文件的内容，若文件太大时，会占用多个block。

ext2/ext3/ext4的日志式文件系统(Journaling filesystem): //ext2本身是没有日志的，在格式化的时候加个选项就有了

新建一个文件的过程:

1.先确定使用者对于想新创建文件目录是否具有w与x的权限;

2.根据inode bitmap找到没有使用的inode号，并将文件的权限和属性写入;

3.根据block bitmap找到没有使用的block号码,将文件的实际数据写入block中,且更新inode的

block指向信息;

4.将刚刚写入的inode与block信息同步更新inode bitmap(位图)与block bitmap ,并更新superblock的内容。

文件的不一致(Inconsistent)状态

例如突然断电、 kernel发生错误等。这样可能写入的信息仅有inode table及 datablock而已,最后一个同步更新(superblock)的步骤并没有做完,此时就会发生metadata的内容与实际信息产生不一致(Inconsistent)的情况。

日志文件系统:

修复文件系统: fsck(检查修复ext的Linux的文件系统 -y 默认yes -f强制检查), e2fsck -fy

案例1：系统无法启动:

```
Welcome to CentOS
Starting udev: [OK]
Setting hostname localhost.localdomain: [OK]
Checking filesystems
/dev/vda2: Superblock last mount time (Thu Mar 31 16:28:03 2016,
           now = Fri Feb 21 22:20:09 2014) is in the future.

/dev/vda2: UNEXPECTED INCONSISTENCY: RUN fsck MANUALLY.
           (i.e., without -a or -p options) [FAILED]

*** An error occurred during the file system check.
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
Give root password for maintenance
(or type Control-D to continue): 输入root密码
```

输入root密码后:

```
*** Dropping you to a shell; the system will reboot
*** when you leave the shell.
Give root password for maintenance
(or type Control-D to continue):
Login incorrect.
Give root password for maintenance
(or type Control-D to continue):
[root@localhost ~]# fsck /dev/vda2
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
Superblock last mount time (Thu Mar 31 16:28:03 2016,
           now = Fri Feb 21 22:21:31 2014) is in the future.
Fix<y>? yes ←

/dev/vda2 contains a file system with errors, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

/dev/vda2: ***** FILE SYSTEM WAS MODIFIED *****
/dev/vda2: ***** REBOOT LINUX *****
/dev/vda2: 21803/512064 files (0.2% non-contiguous), 236893/2045696 blocks
[root@localhost ~]# reboot
```

案例二：Read-only file system

如果运行中的服务器的某一个分区出现readonly，导致进程无法写这个分区（比如nginx进程无法写日志文件到此分区，手动测试touch文件到此分区也显示：cannot touch 'xxxx':Read-only file system），该怎么办？

解决：磁盘read-only的原因一般有2种，一种是没有正常关机导致，还有一种是硬盘故障导致。

如果是/分区，这种情况只能下线报修磁盘了。

如果是其它分区，则可以尝试三步解决此问题：

1. 先卸载此分区

2. 再fsck.ext4 -fy /dev/sdb1

3. 最后挂载此分区，检查是否可以正常读写。

如果仍旧不可以正常读写，请报修磁盘。

注：以上操作过程，请保证服务器不在线上提供服务。

案例三：修复superblock

找到备份的superblock

```
[root@tianyun ~]# dumpe2fs -h /dev/datavg/lv1 |grep 'Blocks per group'
```

```
dumpe2fs 1.42.9 (28-Dec-2013)
```

```
Blocks per group: 8192
```

利用备份的superblock恢复superblock

```
[root@tianyun ~]# fsck.ext4 -b 8192 /dev/datavg/lv1
```

-b指定超块。

查看文件系统信息：

```
dumpe2fs /dev/XXX //查看ext4的文件系统信息
```

```
xfs_info /dev/XXX //查看xfs的文件系统信息
```

修复xfs文件系统：

```
xfs_repair /dev/XXX
```

修复xfs系统要先卸载掉挂载(unmount)，再进行修复。(根目录不能卸载挂载，可以用LiveCD方法)

磁盘空间占满有两种可能：

1.inode号用尽； 2.block用尽。

PS：文件名字不存在inode和datablocks中，存在上级目录的block中，所以删除，创建文件跟文件的权限无关，与上级目录的权限有关。例：因为只有通过访问上级目录才能得到其目录下的inode号。

大部分文件系统都有inode号，fat文件系统没有inode号。

创建文件系统vfat mkfs.vfat /dev/XXX //Linux不识别ntfs分区，Windows不识别ext分区，u盘格式化成fat32Linux可以识别，在Linux里格式化成vfat，Windows也可以识别。

文件链接(软链接， 符号链接)

硬链接：

```

[root@localhost ~]# echo "xxx" > /etc/file1
[root@localhost ~]# cat /etc/file1
xxx
[root@localhost ~]# ll -i /etc/file1 //查看file1的inode号 130624
130624 -rw-r--r--. 1 root root 4 10月 6 20:47 /etc/file1
[root@localhost ~]# ln /etc/file1 /etc/file2 //硬链接 将file1的信息链接到
file2中
[root@localhost ~]# ll -i /etc/file2
130624 -rw-r--r--. 2 root root 4 10月 6 20:47 /etc/file2
[root@localhost ~]# ll -i /etc/file1 /etc/file2
130624 -rw-r--r--. 2 root root 4 10月 6 20:47 /etc/file1
130624 -rw-r--r--. 2 root root 4 10月 6 20:47 /etc/file2 //2为硬链接的数
只有硬链接才有链接数，软链接没有
//file1和file2的inode号相同      链接不算拷贝

```

ln (link files) 命令是一个非常重要命令，它的功能是为某一个文件在另外一个位置建立一个同步的链接。

当我们需要在不同的目录，用到相同的文件时，我们不需要在每一个需要的目录下都放一个必须相同的文件，我们只要在某个固定的目录，放上该文件，然后在其它的目录下用ln命令链接（link）它就可以，不必重复的占用磁盘空间。

Linux文件系统中，有所谓的链接(link)，我们可以将其视为档案的别名，而链接又可分为两种：硬链接(hard link)与软链接(symbolic link)，硬链接的意思是一个档案可以有多个名称，而软链接的方式则是产生一个特殊的档案，该档案的内容是指向另一个档案的位置。硬链接是存在同一个文件系统中，而软链接却可以跨越不同的文件系统。

不论是硬链接或软链接都不会将原本的档案复制一份，只会占用非常少量的磁盘空间。

软链接：-s 创建软链接 在创建的时候如果不加文件名，系统会自动创建跟链接文件相同的名字
//软链接里记录的为源文件的绝对路径，路径有多长，软链接的文件大小就有多大

- 1.软链接，以路径的形式存在。类似于Windows操作系统中的快捷方式
- 2.软链接可以跨文件系统，硬链接不可以
- 3.软链接可以对一个不存在的文件名进行链接
- 4.软链接可以对目录进行链接

硬链接：相当于一个文件可以用多个路径访问，同一个inode号指的为同一个文件 //只有在同一个分区才能做硬链接，目录不能做硬链接。

- 1.硬链接，以文件副本的形式存在。但不占用实际空间。
- 2.不允许给目录创建硬链接
- 3.硬链接只有在同一个文件系统中才能创建

选项：-b 删除，覆盖以前建立的链接 -d 允许超级用户制作目录的硬链接 -f 强制执行 -n 把符号链接视为一般目录 -s 软链接(符号链接) -v 显示详细的处理过程

//链接文件颜色为浅蓝色

任何命令前都可加time 统计其所用的时间。

watch -n1 +'命令' //每隔1秒钟使用一次这个命令 -n后接更新的秒数，命令用单引号引起，否则遇到管道会报错。//任何查看命令都可用。

磁盘阵列 RAID

RAID：廉价磁盘冗余阵列(Redundant Array of Independent Disks)

作用：容错，提升读写速率(I/O速率)

RAID的实现方式：

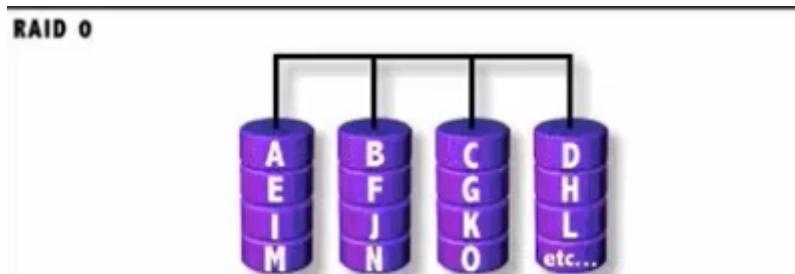
硬RAID：需要RAID卡，有自己的CPU，处理速度快，raid卡分为有电池和无电池。 //硬件RAID

软RAID：通过操作系统实现，比如：Windows，Linux

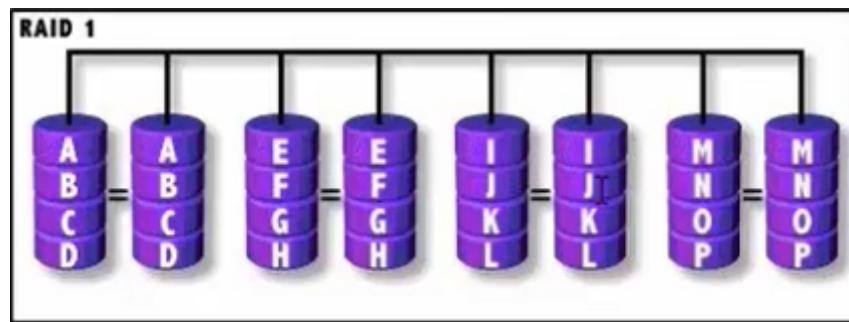
RAID类型	个数	利用率	优缺点
RAID0 条带集	2 +	100%	读写速率快，不容错
RAID1 镜像集	2	50%	读写速率一般，容错
RAID5 带奇偶校验条带集	3 +	(n-1) /n	读写速率快，容错，允许坏一块
RAID6 带奇偶校验条带集双校验	4 +	(n-2) /n	读写快，容错，允许坏两块
RAID01			
RAID10 RAID1的安全 + RAID0的高速	4	50%	读写速率快，容错
RAID50 RAID5的安全 + RAID0的高速	6	(n-2) /n	读写速率快，容错
RAID60 RAID6的安全 + RAID0的高速	8	(n-4) /n	读写速率快，容错

个数为做RAID级别最低磁盘数量，例：RAID0至少要2块磁盘才能做RAID0

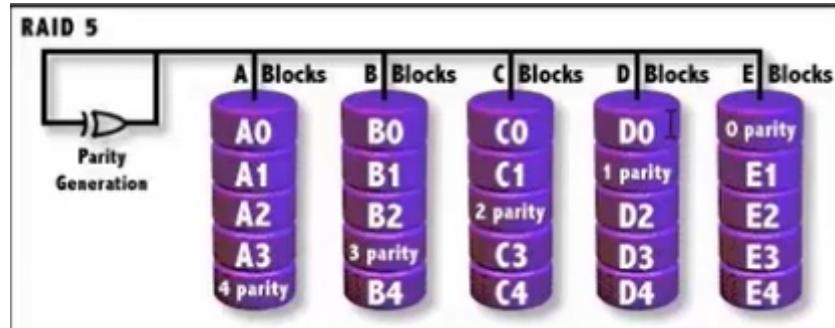
RAID0 个数越多，I/O速率越快 读写速率最快



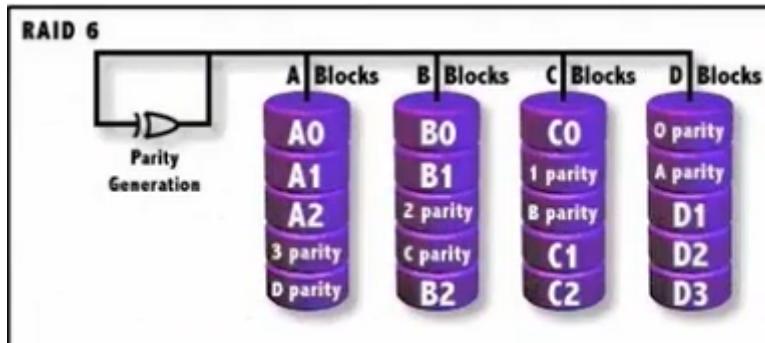
RAID1 将数据分两份存储，一半为数据一半为镜像，读速率快于写速率



RAID5 坏一块盘后，可以通过校验值找回数据。坏一块盘后数据读写会降很多。校验(check)值是根据前面一行所有的数据算出来的。校验值会在每个盘里都存储一份。



RAID6 双校验(dp double parity)



创建RAID5:

```
事先准备4块硬盘
yum -y install mdadm //确保mdadm命令可用
mdadm -C /dev/md0 -l5 -n3 -x1 /dev/sd{d,e,f,g} //dev/xxx这里以d,e,f,g为例
-C 创建RAID      /dev/md0 第一个RAID设备 //RAID设备一般以md命名      -l5
RAID5 //RAID级别      -n  RAID成员的数量      -x  热备磁盘的数量 // -n -x 后面的四
个盘必须和前边的3+1必须吻合
不打算要热备磁盘 不用加-x参数就行 后边三个硬盘就好
mdadm -D //查看RAID的详细信息  mdadm -D /dev/md0
设置raid开机生效: mdadm -D -s > /etc/mdadm.conf
```

Isof (list open files) 恢复进程打开的文件

Linux(FD文件描述符) Windows(文件句柄) 进程每打开一个文件就会有一个文件描述符(文件句柄)。

yum -y install Isof //装Isof

Isof 查看一个进程打开了哪些进程

例: Isof | grep message //var/log/message(系统的主日志文件)

ll /proc/PID/fd

通过文件描述符查看文件的内容:

less /proc/进程号/fd/文件描述符

重定向恢复误删除的文件(例为主日志文件): less /proc/进程号/fd/文件描述符 > /var/log/message 也可以cp回来 cp /proc/PID/fd/文件描述符 /var/log/messages

若误删除掉文件，但有其他进程打开这个被误删除的文件，可以通过Isof来恢复文件。应因为其文件描述符没有被释放。

配置文件一般是不会被进程一直打开的，所以删除后很难恢复。(.conf配置文件)，进程配置文件不会一直打开，只有在开进程的一瞬间读取到内存中，除非你让进程重启才会读取这个配置文件。一般配置文件修改完之后需要重启就是要让内存重新读取配置文件。

文件查找

1.grep: 对文件内容过滤

2.find: 文件查找, 针对文件名(每次寻找都会在硬盘上遍历)

find [[]][选项] [路径] [表达式] 均为可选项 路径可以为多个, 中间用空格隔开, 例: find /etc /var /mnt wo.txt 表示在etc,var,mnt下寻找wo.txt

直接使用find会列出当前目录下的所有文件

(1)按照文件名找: find /etc -name "wo.txt"

find /etc -iname "wo.txt" //忽略大小写

find /etc -name "wo*"

(2)按文件大小查找: find /etc -size +5M //在etc下查找大于5M的文件

find /etc -size -5M //在etc下查找小于5M的文件

find /etc -size 5M //在etc下查找等于5M的文件 //一般找到之后都自动加个选项 -print 将找到的文件打印出来, 一般是不需要加-print的。 可以加 -ls 将打印出的文件进行处理
find /etc -size -5M -ls

(3)按时间查找(atime,ctime,mtime): find /etc -mtime (+/-)5 //查找修改时间超过/小于/等于5天的文件

(4)按文件属主, 属组查找: find /etc -user yu //查找属主是yu的文件

find /etc -group chao //查找属组是chao的文件

find /etc -user yu -a -group chao //查找属主是yu, 属组是chao的文件

-a 为与的意思, 前后条件满足的文件, find /etc -user yu -group chao 不加-a默认为与, 满足两个条件的文件

-o 为或的意思, 条件满足其一即可。

find /etc -user yu -o -group chao //查找显示属主是yu或属组是chao的文件

find /home -nouser -o nogroup //在home下查找没有属主或属组的文件, 这种文件一般为删除了其属组和属主没有连其目录一起删除的文件

(5)按文件类型查找: find /dev -type f/d/l/b/c/s/p 在dev下查找f普通/d目录/l链接/b块设备/c字符设备/s套接字/p管道文件

(6)按文件权限查找: find . -perm 644 //当前目录, 在当前目录查找权限为644的文件 (只能为rw,r,r权限的文件)

find . -perm -644 // //在当前目录查找权限为包含rw,r,r的文件 (- 包含 只要权限里包含rw,r,r的文件都显示)

(7)按正则表达式寻找: find /etc -regex ".*wo[0-9]"

// .*任意多个字符; [0-9] 任意一个数字

找到之后的处理:

-print -ls -delete 找到后删除并不提示 例: find /etc -name wo.txt -delete 查找之后若还需要进行操作就用-exec或-ok -exec(在删除或覆盖之前无提示) -ok(在删除或覆盖之前会有提示和确认) exec和ok后加自定义的shell命令 例: find . -iname wo.txt -exec rm -rf {} \; \; 为固定语法。 {} 表示前边找到的东西; rm -rf {} 为将前边找到的文件删除掉

将/etc/中的所有目录(仅目录)复制到/tmp下, 目录结构不变:

```
find /etc -type d -exec mkdir /tmp/{} \; //查找etc下的所有目录, 并在/tmp/创建  
找到的目录
```

将/etc目录复制到/var/tmp/, 将/var/tmp/etc中的所有目录设置权限为777(仅目录),
将/var/tmp/etc中的所有文件设置权限为666:

```
1. cp -rf /etc /var/tmp/  
    chmod -R a=rwX /var/tmp/etc/ //x仅作用于目录  
2. find /var/tmp/etc/ -type d -exec chmod 777{} \; //一个一个设置  
    find /var/tmp/etc/ -type d -exec chmod 777{} \+ //+统一设置, 效率高于;  
    find /var/tmp/etc/ ! -type d -exec chmod 666{} \; //为/var/tmp/etc/下的所有不  
是目录的文件设置权限为666 ! 加在表达式前表示取反。
```

xargs (英文全拼: eXtended ARGuments) 是给命令传递参数的一个过滤器, 也是组合多个命令的一个工具。之所以用到这个命令, 是由于很多命令不支持|管道来传递参数, 而日常工作中有这个必要, 所以就有了 xargs 命令。

find结合xargs命令:

```
find . -name "wo.txt" | xargs rm -rf
```

```
find /etc -name "wo.txt" | xargs -l {} cp -rf {} /var/tmp //将管道前边的东西赋值给定义的{}, 然  
后将其再拷贝到/var/tmp里
```

3.查找任意文件: locate(查询的数据库: /var/lib/mlocate/mlocate.db) 计划任务: 每天自动更新数据库 /etc/cron.daily/mlocate.cron 手动更新数据库: updatedb 每次增加文件都需要手动更新数据库才能用locate命令查找到, locate命令不适合寻找经常更新的文件。//在找文件之前应先更新数据库

装locate

```
yum -y install locate  
//若提示没有找到该软件包, 利用yum provides locate 查看一下它属于哪个包 然后再用yum -  
y install mlocate 装locate命令
```

4.查找命令文件的路径: which 命令 例: which cd

whereis 命令

PATH变量(环境变量) echo \$PATH 查看变量的值 echo \$变量名

```
[root@localhost yuchao]# echo $PATH  
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/yuchao/.local/bin:/home/yu  
chao/bin //PATH的值
```

命令的路径是通过环境变量PATH的值中找

find: '/run/user/1000/gvfs': 权限不够 查找时报错。

解决方法: umount /run/user/1000/gvfs //卸载该文件

```
rm -rf /run/user/1000/gvfs //删除文件 之后就不会报错了
```

文件打包与压缩

gzip 目录 会将其之下的所有文件压缩但不会打包。

打包最好对目录进行打包，别对文件打包。

tar 本身不具有压缩功能，只具有打包功能，有关压缩及解压是调用其它的功能来完成。

tar -t 查看内容

-r: 向压缩归档文件末尾追加文件

-u: 更新原压缩包中的文件

-C 解压到另一个路径中（备份时用）

例：tar -xf /etc/wo.tar.xz -C /tmp/wo

压缩：

tar -c 表示创建一个打包 -f 表示创建的包名叫啥(参数必须挨着创建的包名) -z 调用gzip -j 调用bzip -J 调用xz 压缩文件的大小xz<bzip<gzip xz的压缩大小最小

例：tar -czf 包名

若不知道文件的类型为什么，可利用file 文件名 查看

解压：

tar -x 表示解压 若不写zj系统会自动判断利用哪个解压工具解压 例：tar xf 包名 -v 显示解压过程

解压ZIP文件：

unzip 文件名.zip

RAR文件可以在Windows中解压，再压缩成zip传到Linux

案例1：mysql物理备份及恢复

```
[root@localhost ~]# tar -czf /backup/mysql.tar.xz /var/lib/mysql  
[root@localhost ~]# rm -rf /var/lib/mysql/*  
[root@localhost ~]# tar -xf /backup/mysql.tar.xz -C /
```

案例2：mysql物理备份及恢复

```
[root@localhost ~]# cd /var/lib/mysql  
[root@localhost mysql]# tar -czf /backup/mysql.tar.xz *  
[root@localhost mysql]# tar -xf /backup/mysql.tar.xz -C /var/lib/mysql
```

host A /etc(海量小文件) -----> host A /tmp

tar -czf - /etc | tar -xf - -C /tmp // -czf后的-表示的为在内存中进行，因为磁盘读写很慢 -xf - -C之前的-表示的是之前在内存中进行的文件解压到/tmp中

host A /etc(海量小文件) -----> host B /tmp

常规方法：scp -r /etc 目标ip:/tmp

建议方法： host B : 监听端口

firewall-cmd --peremanent --add-port=随便一个端口号/tcp

freewall-cmd -reload //重载防火墙

或关闭防火墙 systemctl stop firewalld.service //关闭防火墙

nc -l [-t/-u]随意端口号 | tar -xzf - -C /tmp //nc监听端口，若有数据传送过来会在内存中解压到/tmp下 -t和-u为可选项，不加的话默认为-t -t TCP -u UDP

host A: A,B主机的端口号必须相同

tar -czf - /etc | nc 目标ip地址 目标监听的端口号 //在内存中压缩，并通过端口访问到目标ip地址的主机中

软件包管理

软件类型：1.源码包 需要编译

2.二进制包 已编译

常见二进制包：(不管是源码包，还是二进制包安装时都可能会有依赖关系)

系统平台	包类型	工具	在线安装(自动解决依赖关系)
Redhat/Centos	RPM	rpm,rpmbuild	yum
Ubuntu/Debian	DPKG	dpkg	apt

rpm包：软件包名-版本号(version)-发布版本(release)-系统平台

YUM管理源：

```
yum repolist //repository 仓库 list 列表    列出当前系统有的YUM源仓库列表  
默认有三个仓库(base(基本),extras(扩展),updates(升级))  
yum clean all //清除原来旧的YUM数据库信息  
yum makecache //重新更新缓存，更新新的YUM仓库信息
```

```
mkdir /etc/yum.repos.d/beifen //创建一个备份目录  
mv /etc/yum.repos.d/{*.repo,beifen} //将目录里原本的yum源移到beifen里  
wget -O/-P /etc/yum.repos.d/xxx.repo YUM源网址 // -O 将文件写入FILE里， -P下载到  
指定目录 若没有wget命令可以用curl -o进行下载  
yum clean all //清除所有的包，包括下载的包和缓存的包  
yum makecache //更换YUM源后的操作  
开源镜像网站: mirrors.163.com      mirrors.aliyun.com
```

(上边的源基本都是系统基础源，里面的软件差不多都是和系统有关的文件)

EPEL：(红帽扩展源)

yum -y install epel-release(国外源)

或

```
wget -O /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-  
7.repo //阿里源
```

EPEL里的软件有可能版本会比较旧，可以去软件官方源，然后做一个repo文件做源，装软件。(在官方里找帮助文档(documentation)查看具体如何使用yum repository 字眼安装软件yum源)

YUM管理RPM软件包：

yum的步骤为先下载并安装

yum list 包名 //查看包是否安装了 源前边加了 @ 代表已装

yum -y 在装的时候会连依赖包一起装了

```
yum -y remove 包 //卸载包
```

若要卸载一个包，只会卸载此包，不会卸载其的依赖包。

```
yum list //不加参数，会列出所有包(已装和未装)
```

可以配合grep来实现，不知道具体包名的包 yum list|grep htt

yum -y reinstall 包名 //重新再装一遍包，因为如果之前安装的包是最新版本的，系统会提示为最新版本而无法重装，用reinstall可以实现重装。用于该包缺少一些文件时，又不想卸载，可以直接 reinstall。

```
yum -y update //若不加参数，它会将所有软件包括内核对都更新一遍。
```

```
yum -y update 包名 //更新指定的包
```

```
yum -y update kernel //升级内核
```

升级内核需要重启才会实现。 uname -r //查看内核版本 uname -m //查看系统的架构
x86/x64 uname -a //查看系统所有信息

从本地安装下载的包 yum -y install 本地包的路径 //例: yum -y install /root/httpd.rpm //用 yum装本地包，有一个好处是在装的过程中若有依赖包会自行下载

```
yum -y install 包名 --downloadonly //仅仅只是下载包，而不安装
```

```
yum -y groupinfo 组名 //查看组所属的包
```

```
yum -y groupinstall 组名 //安装这个组的所有包
```

```
yum -y groupremove 组名 //移除组包
```

查询包(从本地，rpm数据库，yum源中查询):

```
yum list httpd php ftp //查询httpd，php与ftp包
```

```
yum list * vnc * //查询列出含有vnc的所有包
```

```
yum list |grep vnc //过滤含vnc的包和上边效果差不多
```

yum list installed //列出已经装过的所有包 //Linux装过的所有rpm包(源码包不会)的信息都会放入rpm数据库中，没有装过的可以在yum源中查询到，Windows装过的所有软件会放入注册表中

yum list installed 包名 //在本地rpm数据库中查询有没有装过该包 可以用echo \$? 显示 装过为非0，没装过为0 \$ 后跟变量 ? 为变量 yum list installed httpd &>/dev/null; echo \$?//查看本地rpm数据库有装过httpd没，输出结果给了null，装过为非0，没装过为0 yum list installed httpd &>/dev/null && echo "装过" || echo "没有装过" //若所指的包装过执行"装过",否则执行"没有装过"

```
yum info 包名 //查看包的信息(名字，架构，版本等) 例: yum info httpd
```

yum group list=yum grouplist //列出所有可用的软件组 例: yum -y groupinstall 软件组名 (例:NFS file server) //这样是错误的会认为是三个组 yum -y groupinstall " NFS file server" //这样才是正确的，""包起来代表一个组

卸载包:

```
yum -y remover 包名
```

```
yum -y groupremove 软件组名 //以组的形式卸载组里所有的包
```

```
yum history //查看装过文件的历史 有编号
```

```
yum history info 编号 //查看编号为某的具体历史信息
```

yum history undo 编号 //undo 撤销，卸载编号为某的软件包，连其依赖包一起卸载掉，卸载掉的包也会在history里有一个新的编号，若想再恢复，再次使用yum history undo 新编号 //将原先撤销掉的包再撤销(恢复)。

扩展查询：

yum list |grep chinese //只是过滤软件包的名字，只关注包名是否含有关键字

yum search chinese //关注包名和描述(介绍) 查询多个字符可用用""引起来，代表是一个字符
"web server" != web server

yum provides 包名/路径 //查看包是由哪个源提供的(可以查看具体的名字和版本)或由属于哪个包提供的 若只记得配置文件名，可以用 yum provides */vsftpd.conf //以 *代表忘记的路径，提供的路径越详细，查的包越少 yum provides */vsftpd/vsftpd.conf 忘记的东西以 * 代替就好

查看命令是由哪个包提供的： yum provides 命令 //查文件需要提供路径，命令不需要
yum的缓存路径 /var/cache/yum/x86_64/7/base(updates,extras)/packages/

自建yum源

建立YUM服务器：

1.配置防火墙：

firewall-cmd --permanent --add-service=ftp //防火墙打开ftp访问，也可以直接把防火墙关了
firewall-cmd --reload

2.关闭SELinux：

0.1

setenforce 0

vim /etc/sysconfig/selinux

SELINUX=disabled

0.2

sed -ri '/^SELINUX=/cSELINUX=disabled'

3.FTP：

yum -y install vsftpd

systemctl start vsftpd //启动ftp服务

systemctl enable vsftpd //开机自启

打开yum的缓存功能：

```
[root@localhost ~]# vi /etc/yum.conf
[main]
cachedir=/var/cache/yum/$basearch/$releasever    //缓存的目录
keepcache=0  //打开yum源的缓存0关闭，1打开 默认为不缓存
```

一、提供update软件包(yum缓存) 源的后缀必须为.repo名字随意起

yum -y install createrepo //创建yum源的工具

先打开缓存，然后清空缓存目录yum clean all，然后yum -y update //自己既有缓存又更新自己所有的包，然后共享缓存的包(把缓存目录的包，移动到要共享的目录中 例： /var/ftp/update)
find /var/cache/yum/x86_64/7/ -iname "*.rpm" -exec cp -rf {} /var/ftp/update/ \; //
在/var/cache/yum/x86_64/7/查找名字以.rpm结尾的所有包，并拷贝到/var/ftp/update(要共享的目录)里；createrepo /var/ftp/update //自动将/var/ftp/update里的包做成yum源，并在目录里创建一个目录repodata(yum数据库文件)

配置nginx源(配置其它软件的源方法都一样)：

访问nginx官网，复制nginx源的路径等，粘贴到本地 vim /etc/yum.repos.d/nginx.repo中,清理缓存(yum clean all),yum install nginx --downloadonly (因为要建的是共享给其它PC的源，所以我们不用装),创建好要共享的目录，mkdir /var/ftp/nginx,find /var/cache/yum/x86_64/7/ -iname "*.rpm" -exec cp -rf {} /var/ftp/nginx/ \;，最后创建repodata，createrepo /var/ftp/nginx //如果加入新软件包，重新创建

配置nginx源--->downloadonly---->创建repodata

vi /etc/yum.repos.d/源名.repo //添加yum源

yum源的构成：

[yum源的ID自己取]

name=自己取

baseurl=源里包的路径 例： ftp://XXX.XXX.XXX.XXX/update //其他地址也行(例： http也行)，因为上边的例子里路径是访问的ftp里的update 路径必须对，不然无法正常访问

gpgcheck=0 //签名检查功能

enabled=1

提供基础软件Base：

挂载centos镜像：

mkdir /var/ftp/centos7 //创建一个能挂载的目录，这是举例

mount 镜像的路径(XXX.iso) 挂载目录的路径(例： /var/ftp/centos7)

echo "mount 镜像的路径(XXX.iso) 挂载目录的路径(例： /var/ftp/centos7)" >> /etc/rc.local //实现开机自动挂载

chmod +x /etc/rc.d/rc.local

挂载光盘：(临时)

mount /dev/cdrom /media 或 mount 光盘名 /media //cdrom链接文件，镜像的文件 可以将其挂载到其它目录上

添加yum源：

vi /etc/yum.repos.d/XXX.repo

[centos7]

name=dvd

baseurl=file:///media //file://链接本地目录 /media 根目录下的media

gpgcheck=0

enabled=1

```
yum update --downonly //下载包，但不安装
```

YUM签名机制：

gpgcheck=0/1 //是否开启签名检查 1开启, 0不开启 //gpg为加密的方式

gpgkey=签名公钥的路径

在装包时： yum -y install 包名 --nogpgcheck //无论配置文件中签名检查开关都不会影响它不检查签名。 --nogpgcheck //不检查软件包的签名

私钥签名，公钥验证

YUM 签名检查机制

=====

rpm软件提供组织例如redhat在构建rpm包时，使用其私钥（private key）对rpm进行签名
client在使用其rpm包时，为了验证其合法性，可以使用redhat提供的公钥（public key）进行签名检查

方法一：事先导入公钥

```
[root@tianyun ~]# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7      //redhat    I
[root@tianyun ~]# vim /etc/yum.repos.d/CentOS-Base.repo
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=\$releasever&arch=\$basearch&repo=os&infra=\$infra
#baseurl=http://mirror.centos.org/centos/\$releasever/os/\$basearch/
gpgcheck=1
```

方法二：指定公钥的位置

```
[root@tianyun ~]# vim /etc/yum.repos.d/CentOS-Base.repo
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=\$releasever&arch=\$basearch&repo=os&infra=\$infra
#baseurl=http://mirror.centos.org/centos/\$releasever/os/\$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

使用rpm工具管理rpm包：(rpm装包需要考虑依赖包)

需要考虑的问题：

1.OS版本 //cat /etc/redhat-release 2.系统架构 //uname -m 3.依赖关系 4.rpm包的版本

rpm只是包管理方式，rpm包有可能为二进制也有可能为源码包。

获得rpm包： rpmfind.net pkgs.org(比前一个网站人性化)

ldconfig 配置动态链接库 -p 打印当前已经加载完的缓存(显示所有的缓存) 直接ldconfig为加载动态链接库

rpm -e 包名 //卸载指定的包

--force //强制安装软件包 相当于 yum reinstall

--nodeps //忽略依赖关系 若要强制安装和卸载有依赖关系的包，应该用--nodeps而不是force

--nosignature //不检查软件包的签名

rpm安装：

rpm -i 显示套件的相关信息。

-v 显示指令执行过程。

-h或--hash 套件安装时列出标记。

-U<套件档>或--upgrade<套件档> 升级指定的套件档。

一般安装rpm用 rpm -ivh 包路径(包名+版本号+发布版本+架构)就好 若要升级 rpm -Uvh 包路径 RPM包安装信息都会保存到RPM数据库中。

rpm查询：只能从rpm本地数据库中查询(只有装过的包才能查询), yum可以在本地数据库和yum 源中查询

rpm -q 包名 //查询指定包是否安装

-qa //查询装过的所有包

-ql 包名 //查询指定包安装的文件

-qf 文件路径 //查询指定的文件属于哪个rpm包

-qi 包名 //查询包的information

-qc 包名 //查询某个包安装的配置文件

-qd 包名 //查询指定包安装的帮助文档

查询没装过的包的信息：(从套件中查询)

rpm -qip 包的路径 // -p 后跟路径

rpm卸载：

rpm -e 包名 //卸载指定包

rpm -e 包名 --nodeps //卸载有依赖的包

源码包管理：

获取源码包：apache, nginx....的官方网站例找 //stable 稳定版本 mainline 主线版(非稳定版本)

准备工作：编译环境：gcc,make //以nginx为例

解压后包里的src为编写的c程序，开发者写的configure来帮助使用者编译。

pcre 支持正则 compiler 编译器

源码安装三部曲：

要安装源码包，得先进入源码包所在的目录里，任何操作在目录里

#./configure // ./代表的是相对路径 因为执行命令的时候\$PATH 会查看绝对路径是否有指定文件，一查肯定没有，会显示没有找到文件。 //执行成功后生成makefile文件

a.指定安装路径 例：--prefix=/usr/local/nginx //prefix 前缀 suffix 后缀 (学习的时候装的路径，项目不往这里装)

b.启用或禁用某项功能 例：--enable-ssl,--disable-filter,--with-http_ssl_module //启用ssl,禁用filter等

c.和其他软件关联 例：--with-pcre

d.检查安装环境，例如是否有编译器gcc，是否满足软件的依赖需求

最终生成:Makefile

#make //按Makefile文件编译，可以使用-j 数字 指定几颗CPU编译，优化编译器参数 lscpu查看cpu 直接make默认认为一个cpu编译 //编译完的可执行程序会出现在当前目录中

#make install //按Makefile定义的文件路径安装，将一些文件拷贝到预先定好的路径里

装好之后启动：/usr/local/nginx/sbin/nginx

```
ps aux |grep nginx //查看启动效果  
ss -tnlp |grep :端口号 //查看指定端口号是否开启  
利用PID可以查看它启动了哪些文件 例: PID为11121 ll /proc/11121/fd
```

报错: error: C compil cc is not found //缺少c编译环境 下载gcc yum -y install gcc-c++
(make)

error:the HTTP rewrite module requires the PCRE library 缺少PCRE库 查找PCRE库
ldconfig -p |grep -i pcre 一般缺少XXX库直接装XXX-devel库就行 这里缺少PCRE yum -y
install pcre-devel yum -y install XXX-devel

error:the HTTP gzip module requires the zlib library gzip模块需要装zlib库 装zlib库就行
yum -y install zlib-devel

error:SSL module requires the OpenSSL library 缺少ssl库 yum -y install open-ssl-devel

emerg紧急报错 getpwnam("XXX") failed //XXX用户没有找到 useradd XXX 创建一个用户就好

安装源码的帮助: README 下载完的包里一般会有

用YUM装的时候遇到问题:

Another app is currently holding the yum lock; waiting for it to exit...

另一个应用程序是: PackageKit

内存: 138 M RSS (481 MB VSZ)

已启动: Tue Dec 1 22:12:55 2020 - 10:15之前

状态 : 睡眠中, 进程ID: 3441

解决方法: rm -rf /var/run/yum.pid

计划任务:

计划任务: 计划任务主要是做一些周期性的任务, 目前最主要的用途是定期备份数据。

at 一次性调度执行

cron 循环调度执行

所有的计划任务执行中的输出都会以邮件的方式发送给指定用户, 除非重定向。

yum -y install at systemctl start/restart/status atd //开启/重启/查看状态atd systemctl
enable atd //开机自启 atd为at的守护进程, 若没有atd, 写的at的计划任务是不会运行的。

at 一次性任务创建好后会交给atd来处理。

at用法: at 时间 时间可以为: now +5min teatime tomorrow (teatime时间指的是16:00) noon
+4 days 5pm 3 2000 //几年几月几日 2000年3月5日

atq 查看所有计划任务列表(里面的编号可以用来关闭计划任务)

atrm 删除计划任务 atrm 编号 //删除编号为XXX的计划任务

at 也可以在文件里创建任务, 在文件中写任务, 然后将文件里的任务重定向输入给at。例: at now
+1min < at.job //at.job为自己随意创建的文件, 内容为计划任务 例: touch /etc/wo.txt 上面任务创
建好会在1分钟之后在/etc/创建一个wo.txt文件

//若执行的用户不是root而是其他wheel组里的用户, 可以用visudo打开文件, /wheel搜索wheel组,
把%wheel ALL=(ALL) NOPASSWD:ALL 前边的#注释删除掉, 把%wheel ALL=(ALL) ALL 这行注释(#)掉
就行了, wheel组里的用户以后在执行命令的时候就不需要输入密码了。以wheel组里的用户运行 at
now +1min < at.job at.job里的内容sudo useradd wo 若没有设置上边visudo里的配置是执行不

成功的，因为需要用户的密码 sudo执行的时候需要考虑终端的问题，若出现问题，打开visudo 查找
tty /tty 把Defaults !visiblepw 这行注释(#)掉就行

corn周期性计划任务：用户级和系统级

corn的守护进程cornd

systemctl status/start/restart crond //查看corn的运行情况/启动/重启

cornd进程每分钟会处理一次计划任务。 d代表的是daemon 守护进程

用户级：

计划任务存储的位置：ls /var/spool/cron/ 删除计划任务目录里的邮件：>/var/spool/mail/root //将空重定向到root里相当于删除 或 进到mail里 d 邮件编号 删除指定邮件编号的邮件 例：d 1-30 删除1-30邮件 //每个用户都会有一个和它用户名相同的计划任务

不允许指定用户有计划任务可以编辑 vi /etc/cron.deny文件，将不允许有计划任务的用户添加在其中就好。

crontab -l //列出当前用户的计划任务

crontab -r //删除计划任务

crontab -e //编辑计划任务

管理员可以使用crontab -u username 去管理指定用户的计划任务，普通用户只能管理自己的计划任务。

语法格式：分 时 日 月 星期 命令

分值为*代表(0-59) 时值为 * 代表(0-23) 天值为 * 代表(1-31) 月值为 * 代表(1-12) 周值为 * 代表(0-6) (0和7都代表星期7)(sun,mon,tue,wed,thu,fri,sat 0-6) * * * * * date //计划任务表示间隔每分钟都执行一次date 一般执行的为脚本，而不是命令 脚本需要执行权限

例：0 2 * * * XXX.sh //每天2点执行XXX.sh

0 2 14 * * XXX.sh //每月14号2点执行

0 2 14 2 * XXX.sh //每年2月14号2点执行

0 2 * * 5 XXX.sh //每周5的2点执行

0 2 * 6 5 XXX.sh //每年6月的星期5的2点执行

0 2 2 * * XXX.sh //每月2号的2点或每周5的两点

0 2 2 6 5 XXX.sh //每年6月2号的两点或每周5的两点整执行 //星期和日为或的关系 两个时间点都执行

*/5 * * * * XXX.sh //每隔5分钟执行一次(默认认为隔一分钟执行一次) //斜线放其他后边加数字表示每隔n分钟，n小时，n天，n月，n周

0 2 1,4,6 * * XXX.sh //每月1,4,6号的2点整执行

0 2 1-5 * * XXX.sh //每月1到5号2点整执行

0 * * * * XXX.sh //每天小时整点执行一次

* 02 * * * XXX.sh //每天2点中的每一分钟都执行 //02=2

0 2 * * * XXX.sh //每天2点整执行一次

```
* * 14 2 * ls //每年2月14号的每分钟都执行  
,多会和多会 - 多会到多会 *任何时候 /每隔多会
```

系统级计划任务：(用户计划任务在创建时不需要写明用户，而系统计划任务在创建时要写哪个用户执行)

临时文件的清理：/tmp /var/tmp

系统信息的采集：sar命令 从计划任务的采集里获取的

日志的轮转（切割）

定义计划任务的位置：

1.vi /etc/crontab //该文件默认是没有任何计划任务 例：* * * * * 用户名 执行的命令 不建议在里面写计划任务

2./etc/cron.d/* 0hourly 例：01 * * * * root run-parts /etc/cron.hourly //run-parts表示后边是一个文件夹 表示每小时01分以root身份执行/etc/cron.hourly目录下的所有脚本 cron.hourly 目录下有一个脚本 0anacron 这个脚本的内容/usr/sbin/anacron -s 每小时启动anacron进程

cron 的计划任务执行时间被错过了就不会再执行了

anacron 每小时的01分被crond唤醒。防止被任何故障错过的计划任务

anacron的配置文件为 /etc/anacrontab

```
anacron 配置文件里的内容 //period 执行的周期 单位(天) delay 执行的延迟 单位为  
min(分)  
#period in days delay in minutes job-identifier(标识) command(命令)  
1 5 cron.daily nice run-parts /etc/cron.daily //run-parts 后跟  
目录路径 执行目录下的所有脚本  
7 25 cron.weekly nice run-parts /etc/cron.weekly  
@monthly 45 cron.monthly nice run-parts /etc/cron.monthly  
检查有无执行过：在 /var/spool/anacron/cron.daily(/.weekly/.monthly)里有执行过的时间，  
若没有执行过时间不会更新，然后等待执行延迟过去后执行一次。例如昨天没执行过，在今天8: 01发现昨天  
没有执行，会在8: 06执行一次。
```

```
[root@localhost cron.hourly]# cat /var/spool/anacron/cron.daily  
20201030  
[root@localhost cron.hourly]# cat /var/spool/anacron/cron.weekly  
20201030
```

cron的日志文件 tail -f(或tailf动态查询) /var/log/cron

日志管理

rsyslog 日志管理

logrotate 日志轮转

日志管理大致分为：采集 ----> 分析

一、处理日志进程：

rsyslogd(系统日志的守护进程)：绝大多数日志记录，收集和系统操作有关，安全，认证sshd, su, 计划任务at, cron等

httpd/nginx/mysql:可以以自己的方式记录日志。

查看守护进程: ps aux |grep rsyslog //在/usr/sbin/rsyslogd -n

rsyslogd在采集的时候日志可以存放在本地，也可以存放在远程服务器中。

二、常见的日志文件(系统、进程、应用程序)

tail /var/log/messages //系统主日志文件

tail -f /var/log/messages //动态查看日志文件的尾部 //tailf相当于tail -f

tailf /var/log/secure //认证与安全相关日志

tailf /var/log/maillog //跟邮件postfix相关

tail /var/log/cron //crond,at进程产生的日志

tail /var/log/dmesg //和系统启动相关

tail /var/log/audit/audit.log //系统审计日志(一般不开，耗资源)

tail /var/log/yum.log //yum的日志

tail /var/log/mysqld.log //mysql日志

tail /var/log/xferlog //和访问ftp服务器相关

w //查看当前登录信息

查看记录每个用户的登录次数和持续时间等信息 /var/log/wtmp 文件不能看，二进制文件，用cat 和vi会乱码，用last可以看 last /var/log/wtmp

last //查看现在登录的用户 /var/log/utmp 只能用last看 last /var/log/utmp

lastlog //所有用户的登录情况

案例：统计登录失败top 5

grep 'Fail' /var/log/secure |awk '{print \$11}' |sort |uniq -c |sort -k1 -n -r|head -5

//过滤登录失败的用户，'fail'为日志里的特征 //打印所有失败的用户的ip \$11 为第11列 \$NF为倒数第一列 //sort排序 //un iq 去重 //按照登录最多的人排序 -k1 代表第一列 -n 数值 -r 逆序 //head -5 前5行 //若日志文件被切割可以用日志文件路径加 *；例message主日志被切割过，要查时用 /va/log/message *

三、rsyslog子系统(默认有安装)

rpm -qc rsyslog //查看其配置文件

/etc/logrotate.d/syslog //和日志轮转(切割)有关

/etc/rsyslog.conf //rsyslog的主配置文件

/etc/sysconfig/rsyslog //rsyslogd相关文件

/etc/rsyslog.conf: 告诉rsyslogd进程，哪个设备(facility)，关于哪个级别(level)的信息，以及如何处理。要充当日志服务器需要打开防火墙和日志相应的模块，端口为514。例：打开udp远程日志支持把\$ModLoad imudp 和 \$UDPServerRun514 前的注释去掉

函数syslog() man 3 syslog //这个函数在第三章里

用户层的工具logger，可以写日志，logger -p 可以指定日志的设备类型。

日志级别(level syslogd): 遇到何种情况(正常，错误)才会记录日志

LOG_EMERG 紧急，致命，服务无法继续运行，如配置文件丢失

LOG_ALERT 报警，需要立即处理，如：磁盘空间不足

LOG_CRIT 致命行为

LOG_ERR 错误行为

LOG_WARNING 警告行为

LOG_NOTICE 普通信息

LOG_INFO 标准信息 //一般指INFO级别，指的就是info及以上的级别信息都记录。

LOG_DEBUG 调试信息，排错所需，一般不建议用，正确，错误输出都有。

//级别越往上越高，越危险。

在/etc/rsyslog.conf主配置文件里配置级别的处理文件。 例：local5.* /var/log/自定义名字就行
将local5级别的所有日志信息放到/var/log/XXX里处理。

logrotate 日志轮转(切割)

切割之后即使名字还是原来的，但其inode编号变了。 //进程关注的是inode编号

logrotate 日志轮转(切割):等日志大小达到预定的大小，进行切割，产生新的文件 针对任何日志文件(rsyslog日志，Nginx访问或错误日志...)

日志轮转的路径：/etc/cron.daily/Logrotate 内容里命令为：/usr/sbin/logrotate //轮转在每天计划任务里面，每天执行/usr/sbin/logrotate 这个之后的 -s 后的路径为状态文件(哪个日志在哪个时间轮转过，最近轮转的时间) 再之后为 轮转的规则 规则文件在:/etc/logrotate.conf

修改logrotate的规则文件：vi /etc/logrotate.conf 从上往下依次为： weekly //轮转的周期，一周轮转
rotate 4 //保留4份 create //轮转后创建新文件 dateext //使用日期作为后缀 #compress
//是否进行压缩 include /etc/logrotate.d //包含该目录下的文件 notifempty //空文件不轮转

轮转周期不写默认为跟随全局周期，还有一个条件为最小轮转时文件的大小，例：minsize 1M
rotate 1 为备份1份。

例：轮转例子： /var/log/wtmp{ //对该日志(wtmp)设置轮转的方法，被轮转的日志文件

monthly //一月轮转一次

minsize 1M //最小达到1M才轮转

create 0664 root utmp //轮转后创建新文件，并设置权限

rotate 1 //保留1份

} //若没写的规则以全局为主。

chattr +a //增加a属性，只允许追加

chattr -a //去除只允许追加属性

```
测试时先把/etc/logrotate.d/syslog中的messages注释掉 //这里以messages为例
/var/log/messages{
prerotate    //pre表示运行之前的，运行之前先将a属性去除
    chattr -a /var/log/messages //减去日志文件的a属性，因为有a1属性在，轮转会报错失败。
endscript
daily
    create 0600 root root
missingok    //丢失不提醒
```

```
rotate 5      //保留5份
postrotate   //post 运行之后再将a属性加上
    chattr +a /var/log/messages
endscript
}
sharedscripts //共享脚本
```

awk '{print \$1}' /etc/httpd/logs/scss_log |sort|uniq -c |sort -k1 -nr //打印scss_log第一列，排序，去重，按第一列再进行排序

日志文件被轮转之后，需要重载(重新加载)。

```
/var/log/httpd/*log{
missingok
notifempty
sharedscripts
delaycompress
postrotate
/bin/systemctl reload httpd.service > /dev/null 2>/dev/null ||true //轮转
http日志后，重载http服务
endscript}
```

注意：当轮转之后创建新文件，新文件会有新的inode号，如果希望进程如rsyslog、nginx进程会把新的日志写入新日志文件中，但默认不会！！

应该在日志切割后，告诉rsyslog、nginx进程 reload(重载) 或信号(1 or HUP)

Linux网络配置

NetworkManager管理网络：

网络管理器(NetworkManager)是一个动态网络的控制器与配置系统，用于当网络设备可用时保持设备和连接开启并激活。默认情况下，centos7已装网络管理器，并处于开启状态。

device 设备，物理设备，网卡 例：enp2s0, virbr0(虚拟化接口), team0

connection 连接，逻辑设置，网卡配置 指的是一套具体配置方案

多个网卡配置可以应用到一个device，但同一时间只能启用其中一个网卡配置。好处是：针对一个网络接口，可以设置多个网络连接，比如静态IP和动态IP，再根据需要开启相应的connection。

一个网卡可以有多个配置。

NetworkManager提供的工具：

要用networkmanager的命令必须先开启它：systemctl start/status/stop NetworkManager//开启/查看/关闭NetworkManager状态

nmcli(command line 命令行),nmtui(text user interface文本用户接口),nm-connect-editor //只用nmcli就行，后两个不推荐

nmcli device //查看所有的device，设备，类型，状态，连接配置 nmcli device show //更加详细的查看 nmcli device show 具体的device //例：nmcli device show enp0s20u2

nmcli connection //查看所有的connection连接配置，名字，UUID，类型，device nmcli connection show //更加详细的查看

修改IP方法：1.通过命令行(不推荐)，2.修改配置文件

1.修改ip命令行(不推荐)

```
[root@localhost yu]# nmcli connection add con-name ens00-my autoconnect yes
iface ens00 type ethernet //add增加一个连接配置, con-name 连接配置的名字, 自己取;
autoconnect 是否自动连接yes/no; iface 接口名; type 类型, ethernet以太网。
连接 "ens00-my" (51034be6-a892-4834-8f4e-0157be27b9c9) 已成功添加。//这样配置完是没有IP的
```

```
[root@localhost yu]# nmcli connection add con-name ens11-my autoconnect yes
iface ens11 type ethernet ip4 192.168.1.168/24 gw4 192.168.1.1 //ip4添加ipv4的
ip地址; gw4 添加ipv4的网关
连接 "ens11-my" (5e5dc77a-bcd2-4064-a8ea-34edc50ada25) 已成功添加。
```

```
nmcli connection up ens11-my //开连接, 如果不开启不会更换新配置
nmcli connection delete ens11-my //删除指定配置
```

增加一个连接就是在 /etc/sysconfig/network-scripts/下创建了一个文件

2.修改ip的配置文件

```
vim /etc/sysconfig/network-scripts/ifcfg-ens00-my

TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp //引导协议, IP有两种获取方式, 一种手动设置, 一种DHCP分配
BOOTPROTO=None为手动设置
(BOOTPROTO=None
IPADDR=xxx.xxx.xxx.xxx //手动设置的ip
PREFIX=xx //掩码 24等
//配置多个IP
IPADDR2=xxx.xxx.xxx.xxx
PREFIX=XX)
GATEWAY=XX.XX.XX.XX //网关
DNS1=8.8.8.8
DNS2=114.114.114.114)
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens00-my //连接名
UUID=51034be6-a892-4834-8f4e-0157be27b9c9
DEVICE=ens00 //设备名
ONBOOT=yes //开机是否自启动
//配置无先后顺序, 手动添加ip时要将BOOTPROTO=None, 并添加IP地址(IPADDR)和掩码
(PREFIX), 网关(GATEWAY)等, 可设置多个IP
```

配置完之后要重新加载一下: nmcli connection reload

```
nmcli connection down ens00(接口名(设备名)) //关掉指定的设备
nmcli connection up ens00 //打开指定的设备 重载, down, up都需要操作才会修改ip
远程修改IP的时候可以将down和up写在一行: nmcli connection down ens00;nmcli
connection up ens00
ip r 查看网关    ip a 查看所有ip设置
```

不用NetworkManager配置IP:
修改/etc/sysconfig/network-scripts/对应的配置文件 vi /etc/sysconfig/network-scripts/配置文件
修改完之后直接 systemctl restart network 就会修改配置

有NetworkManager配置IP: 修改配置-->reload-->down;up
没有NetworkManager配置IP: 修改配置文件--->重启network(systemctl restart network)

如果网卡没有配置文件, 可以拷贝一个到/etc/sysconfig/network-scripts/里, 然后删除掉里面的UUID, 不然会冲突, 修改device和name。

修改主机名:

- 1.echo '需要修改的名称' > /etc/hostname
- 2.hostname 需要修改的名称 -->重启生效

域名解析:

Linux本地hosts路径: cat /etc/hosts
getent hosts 域名 //查看hosts能不能帮我解析指定域名
DNS配置路径: vi /etc/resolv.conf
网卡配置文件里的DNS服务器会同时保存到/etc/resolv.conf里

基本网络查看工具:

ip a 除网关外的各种IP信息
ip a show 网卡 //查看具体网卡设备的信息
ip router 查看路由表(到达什么地方下一跳是谁从哪个端口出)
ip neigh 查看邻居的IP和MAC地址, arp缓存表 (访问过本机的记录)
ping 为icmp协议
ping -c2 域名或/IP //Linux默认是一直ping, c2代表ping2次
ip -s link show 具体设备 查看具体网卡接受/传输/丢弃/广播的包的详细情况
traceroute 域名/ip 路由跟踪, 跟踪到达目标主机的每一跳 最多30跳
tracepath 域名/ip 用来追踪并显示报文到达目的主机所经过的路由信息。
ethtool 网卡 //查看网卡更详细的信息, 速率, 连接状态等
http 80/tcp https 443/tcp ssh 22/tcp ftp 21/tcp

ss命令：

- V, --version 程序版本信息
- n, --numeric 不解析服务名称
- r, --resolve 解析主机名
- a, --all 显示所有套接字 (sockets)
- l, --listening 显示监听状态的套接字 (sockets)
- o, --options 显示计时器信息
- e, --extended 显示详细的套接字 (sockets) 信息
- m, --memory 显示套接字 (socket) 的内存使用情况
- p, --processes 显示使用套接字 (socket) 的进程
- i, --info 显示 TCP内部信息
- s, --summary 显示套接字 (socket) 使用概况
- 4, --ipv4 仅显示IPv4的套接字 (sockets)
- 6, --ipv6 仅显示IPv6的套接字 (sockets)
- 0, --packet 显示 PACKET 套接字 (socket)
- t, --tcp 只显示 TCP套接字 (sockets)
- u, --udp 只显示 UCP套接字 (sockets)
- d, --dccp 只显示 DCCP套接字 (sockets)

查看TCP监听状态的信息: -tnlp t只显示tcp, n不解析服务器的名字, l 只看tcp监听的, p监听的进程

-h, --help 帮助信息

查看TCP的所有状态: -atn a 所有状态的套接字, 包括listen。一个服务会创建很多套接字。

State	Recv-Q	Send-Q	Local Address:Port
Peer Address:Port			
LISTEN	0	128	0.0.0.0:888
	0.0.0.0:*		

监听

在这个ip上:某个进程监听的888端口 //

没有写IP的表示80监听在任何接口上

查看UDP的所有状态: ss -anu

watch -n.1 'ss -atn' |grep ':80' //每隔0.1秒用ss-atn并过滤80端口

临时配置网络信息:

ip/netmask: ip addr add dev eth0 IP/掩码 这里以eth0为例, eth0为具体的网卡 用ip a查看

ip addr del dev eth0 IP/掩码 //删除添加的IP

gateway 网关: ip route del default //删除默认网关

ip route add default via 192.168.10.5 添加路由目标是任何地方, 下一跳是192.168.10.5

ip route add 192.168.10.5/24 via 39.97.93.162 添加一条路由, 到达192.168.10.5下一跳为39.97.93.162

修改网卡名称：改为eth0

已经安装centos7

1.修改网卡配置文件

```
mv ifcfg-eno1677728 ifcfg-eth0
```

```
vim ifcfg-eth0 //修改NMAE=eth0 DEVICE=eth0
```

2.GRUB添加内核(kernel)参数

```
vim /etc/sysconfig/grub //GRUB_CMDLINE_LINUX="crashkernel=auto spectre_v2=retpoline rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet" 在这行quiet后空格添加net.ifnames=0
```

然后grub2-mkconfig -o /boot/grub2/grub.cfg

3.reboot

在安装系统时：添加内核参数 net.ifnames=0//将红帽的新命名的方式关掉

在选择Install CentOS 7的时候按Tab键，然后输入net.ifnames=0然后回车就行



FTP Server

ftp使用两个端口: 21(控制端口, 命令端口), 20(数据端口) (主动模式)

21端口: 用来控制用户验证.

20端口: 用来传输数据.

软件包: vsftpd

FTP默认共享目录: /var/ftp

ftp访问: 匿名访问(以ftp访问)(默认只能下载), 系统用户(默认可下载及上传文件) 匿名用户访问的目录在/var/ftp/ 系统用户访问时的目录是自己的家目录/home/用户名/

访问格式:

1.ftp://账号:密码@ip地址/ 例: ftp://admin:123456@192.168.1.1/

2.ftp://ip地址/

yum -y install vsftpd //装ftp服务

centos6:

service vsftpd start/restart //开启/重启服务

chkconfig vsftpd on //设置为开机启动

关闭SELINUX:1.sed -ri '/^SELINUX=/cSELINUX=disabled' /etc/selinux/config

2.setenforce 0

centos6关闭防火墙

1.Iptables (暂时关闭)

iptables -F

service iptables save

centos7:

systemctl start vsftpd //开启服务

systemctl enable vsftpd //设置开机自启

centos7关闭防火墙: centos7默认firewalld/SELinux已关闭, 若未关闭:

firewall-cmd --permanent --add-service=ftp //防火墙放行ftp //--permanent 永久规则

firewa-cmd --reload //重载ftp

关闭SELINUX:

1.sed -ri '/^SELINUX=/cSELINUX=disabled' /etc/selinux/config //重启才会生效 //找以SELINUX开头的, c表示整行都换为SELINUX=disabled

2.setenforce 0 //立即生效

客户端操作ftp下载与上传等:

1.lftp: yum -y install lftp

语法: lftp 用户名:密码@ftp地址:传送端口 (默认21) (也可以先不带用户名登录, 然后在接口界面下用login命令来用指定账号登录, 密码不显示.)

连上ftp后: ls展示ftp的目录 cd切换ftp的目录

lcd切换本地的目录

get 下载指定的文件 mirror 下载指定的目录(会下载到本地的当前目录中) 例: mirror wo/

put 本地文件 上传文件 mirror -R 本地目录名 上传本地目录 例: mirror -R wo/

by 退出

若使用lftp报错: ls: Fatal error: Certificate verification: Not trusted //可能是证书未被信任,在linux上输入命令时提示出来的。

解决方法：编辑/etc/lftp.conf，加入下面内容：

set ssl:verify-certificate no

或直接在lftp命令提示符下输入： set ssl:verify-certificate no 回车。

2.wget

wget ftp://192.168.1.1/hosts -O /var/tmp/wo.txt //将下载的ftp文件保存到指定的文件里，文件必须事先存在.下载文档到指定的文件里

wget ftp://192.168.1.1/hosts -P /var/tmp/ //P可以放到指定目录里，只是下载文件到哪个目录里

wget ftp://192.168.1.1/hosts -o /var/tmp/wo.txt //将文件下载到目录里，并将下载的屏幕输出的信息保存到wo.txt里

wget ftp://192.168.1.1/hosts -q //下载完不会有任何回显

wget -m ftp://192.168.1.1/wo //下载wo目录

3.网页Chrome, Firefox, IE等

修改FTP的配置文件：修改完配置文件后重启ftp服务，让内存读取重新读取ftp配置文件

vim /etc/vsftpd/vsftpd.conf //ftp的配置文件

anonmous_enable=YES/NO 允许/拒绝匿名用户访问

Local_enable=YES/NO 允许/拒绝本地用户访问

write_enable=YES/NO //是否允许写 (全局=ftp允许写+文件允许写)

local_umask=022 //控制本地用户上传文件的默认权限，umask表示要减掉的权限

anon_umask=077 //控制匿名用户上传文件的默认权限

anon_max_rate=50000 //匿名用户限速 限速多少K 多少+000 例：限速50K 50000

local_max_rate=80000 //本地用户限速

max_clients=500 //ftp最大连接数

max_per_ip=2 //单个IP最大连接数，线程数

local_root=/ftproot //指定本地用户访问的root目录

anon_root=/anonroot //指定匿名用户访问的root目录

chroot: 锁定本地用户HOME

方法一：部分用户chroot

chroot_list_enable=YES

chroot_list_file=/etc/vsftpd/chroot_list //锁定指定列表里的用户

方法二：所有本地用户chroot

chroot_local_user=YES

要使用chroot锁定用户，必须要先把登录ftp用户的家目录的w权限去掉才能正常访问ftp。 //chmod a-w /home/wo 例：wo本地用户要访问ftp,并且这个用户被锁定 //用用户访问ftp的时候会访问到用户自己的家(/home/用户/)里

在修改ftp配置文件时要小心，ftp对语法严格，要注意空格等，若有空格可在vi中用set list显示一下控制字符，看有无多出空格或tab。

NAS

NAS(Network Attached Storage)，网络附属存储。

它是一种专用数据存储服务器。它以数据为中心，将存储设备与服务器彻底分离，集中管理数据，从而释放带宽、提高性能、降低总拥有成本、保护投资。其成本远远低于使用服务器存储，而效率却远远高于后者。

NAS是一个设备，一个功能。而CIFS/NFS是一种协议。可以在NAS上启用CIFS/NFS协议。

NAS之NFS

NFS的客户端主要是Linux。

NFS(网络文件系统 network file system) 存储设备 共享的存储设备为其它服务器提供同样的存储服务 (分流负载均衡) 例：Google服务全球用户，对于同一份数据，而有百万用户访问，服务器肯定撑不住，而利用NFS可以让用户通过访问网站服务而访问同样的数据。

NFS (Network File System) 即网络文件系统，是FreeBSD支持的文件系统中的一种，它允许网络中的计算机之间通过TCP/IP网络共享资源。在NFS的应用中，本地NFS的客户端应用可以透明地读写位于远端NFS服务器上的文件，就像访问本地文件一样。 它可以透过网络，让不同的主机、不同的操作系统可以共享存储。

RPC和NFS的关系：NFS是一个文件系统，而RPC是负责信息的传输。 NFS在文件传送或信息传送过程中依赖于RPC协议。

只要用到NFS的地方都要启动RPC服务，不论是NFS SERVER或者NFS CLIENT。

NFS支持多节点同时挂载以及并发写入。

一、准备存储端(提供存储服务)

1.关闭防火墙 systemctl stop firewalld

```
systemctl disable firewalld
```

2.下载服务 yum -y install nfs-utils

3.创建共享目录 例：mkdir /gongxiang

```
echo "web">>/gongxiang/index.html(前提是已经装了http服务)
```

4.配置nfs配置文件 /etc/exports(这个文件刚开始是空文件)

配置格式为：NFS要共享目录 NFS客户端地址1(共享给谁)(参数1,参数2,参数3.....) 客户端地址2(共享给客户端的权限)

例：/gongxiang 192.168.1.0/24(rw,sync,no_root_squash) //不压制root(当client端使用root挂载时，也有root权限,如果为root_squash的话即使客户端以root身份挂载的也没有写权限) // 将/gongxiang目录 共享给192.168.1.0网段

5.启动服务 systemctl start nfs-server

```
systemctl enable nfs-server
```

exportfs -v 查看共享目录 -a 全部挂载或者全部卸载 -r 重新挂载 -u 卸载某一个目录

二、客户端(前提是已经装了http服务)

以web为例: yum -y install nfs-utils httpd

```
systemctl start httpd
```

```
systemctl enable httpd
```

1.查看存储端共享情况(查看指定IP的存储有无共享)(可选)

```
showmount -e 共享服务的IP地址
```

2.手动挂载 mount -t nfs 共享服务的IP地址:/gongxiang /var/www/html/ //nfs 也是种文件系统
(网络文件系统)

```
umount /var/www/html/ //取消挂载
```

3.自动挂载到网站主目录

```
vim /etc/fstab
```

```
共享服务的IP地址:/gongxiang /var/www/html nfs defaults 0 0
```

然后退出vi mount -a的意思是将/etc/fstab的所有内容重新加载。 因为mount的路径是在/etc/fstab里。

4.查看挂载(可选) df

Windows挂载nfs: (Windows家庭版无, 专业版有)

开始->控制面板->程序->启动或关闭Windows功能->NFS服务

打开服务之后: 打开cmd

输入: mount \\共享的存储设备IP\共享的目录 卷标(不能用已经存在的卷标):

例: mount \\192.168.241.128\gongxiang x:

用已经存在的卷标会报错: 网络错误 - 85 有关详细信息, 请键入“NET HELPMSG 85”。

NAS之CIFS

CIFS的客户端主要是Windows。支持多节点同时挂载以及写入并发。

CIFS 通用的Internet文件系统。由微软推出, 采用C/S模式, 基本网络协议: TCP/IP和IPX/SPX。

```
yum -y install cifs-utils //装cifs内核模块, 不装的话识别不出来cifs
```

一、samba服务器(存储端)

1.安装软件: yum -y install samba

2.建立共享目录: mkdir /gongxiang

chmod 777 /gongxiang //为了让客户端往进写数据 samba服务开启写, 用户也得对文件有写权限才能对文件写

```
cp -rf /etc/hosts /data/
```

3.创建用户: useradd alice -s nologin

```
smbpasswd -a alice //给用户alice设置samba密码
```

4.通过samba共享 //在samba里#和;都是注释

vim /etc/samba/smb.conf //前面的共享不用管, 直接拉最后进行添加 配置文件写共享名, 共享路径, 和写权限就能正常访问samba了

```
[data] //共享名  
path = /data //共享路径  
;valid user = alice //有效用户，只有指定用户可以访问  
;hosts allow = XXX.XXX.XXX. //为注释 //哪个网络可以访问 只写到网络就好，不能写主机位  
例：192.168.1.  
;write list = //指定的XXX可以写，这个跟下边writab冲突，只写一个就好  
writable = yes //让samba用户都可写
```

5.启动samba

```
systemctl start nmb smb
```

```
systemctl enable nmb smb
```

6.firewalld

```
firewall-cmd --permanent --add-service=samba
```

```
firewall-cmd --permanent --add-service=samba-client
```

```
firewall-cmd --permanent --add-service=mountd
```

```
firewall-cmd --reload //重载防火墙
```

客户端连接：

Windows端：

1.win+r \\samba服务器的ip地址

输入用户名和密码即可

2.映射网络驱动器(相当于挂载，在用户登录时启动，相当于开机自启)

Linux端：

```
yum -y install samba-client cifs-utils
```

Linux端访问需要装samba-client服务和cifs-utils服务

1.查看存储共享(可选)

```
smbclient -L 提供samba的服务器IP地址
```

2.手动挂载(可选)

```
mkdir /mnt/samba
```

```
mount -t cifs -o user=alice,pass=密码 //提供samba的服务器IP地址/data /mnt/samba //提供  
samba的服务器IP地址的路径 本地挂载目录
```

3.自动挂载到指定目录

```
vim /etc/fstab
```

```
//提供samba的服务器IP地址/data /mnt/samba cifs user=alice,pass=密码 0 0
```

退出vi mount -a

Windows文件Linux共享文件夹：(linux事先要装cifs-utils 和samba-client服务， yum装)

先右键要共享的文件夹，属性，共享，高级共享，共享此文件夹，起个共享名(最好是英文的),在共享界面看一下网络路径，然后在Linux中mount -t cifs -o user=xy,pass=123 //192.168.241.1/file /mnt/mk/

mount -t cifs -o user=Windows用户,pass=用户密码 //Windows的ip地址/共享的文件夹 /Linux要挂载的路径

挂载时报错：

mount error(13): Permission denied(拒绝访问) //可能是Windows的原因 解决方法：win+r 输入 gpedit.msc Windows设置-安全设置-本地策略-安全选项 找到网络访问:本地账户的共享和安全模型 换成经典-对本地用户进行身份验证...

Windows访问samba时报错：

一、\\192.168.241.128无法访问.你可能没有权限使用网络资源.请与这台服务器的管理员联系?

1.计算机管理窗口，在列表中，选择本地用户和组选项，双击展开。

2.双击展开后，在子项里选择用户，并在右侧双击打开来宾账户，关闭账户已禁用选项。

3.使用快捷键win+r打开运行命令框，在运行命令框中输入secpol.msc命令，来启动本地策略组编辑器。

4.进入到本地策略组编辑器后，依次展开本地策略-用户权分配。展开后，在右边策略窗口中，找到拒绝本地登录，并双击打开。双击打开后，选择来宾账户，点击删除。

5.点击确定好，点击安全选项，在右侧的策略窗口中，双击打开本地账户的共享和安全模型。双击打开后，设置为仅来宾。

6.删除“拒绝从网络上访问这台计算机”项中的guest账户：运行组策略 (gpedit.msc) \计算机配置\windows设置\安全设置\本地策略\用户权利指派\拒绝从网络访问这台计算机。如果其中有guest，则将其删除。（原因是：有时xp的guest是不允许访问共享的）

二、你不能访问此共享文件夹,因为你组织的安全策略阻止未经身份验证的来宾访问?

1、首先按window+R键打开运行键入gpedit.msc启动本地组策略编辑器。

2、在组策略编辑器中找到“计算机配置”-“管理模板”-“网络”-“Lanman工作站”

3、“Lanman工作站”这个节点，在右侧内容区可以看到“启用不安全的来宾登录”这一条策略设置。状态是“未配置”。双击“启用不安全的来宾登录”这一条策略设置，将其状态修改为“已启用”并单击确定按钮。

DNS

名字解析：

NetBios名 网络基本输入输出系统名(这种命名方式只适用于局域网) 由WINS服务提供解析 本地还有hosts文件

FQDN：完全限定域名 www.baidu.com baidu.com mail.qq.com

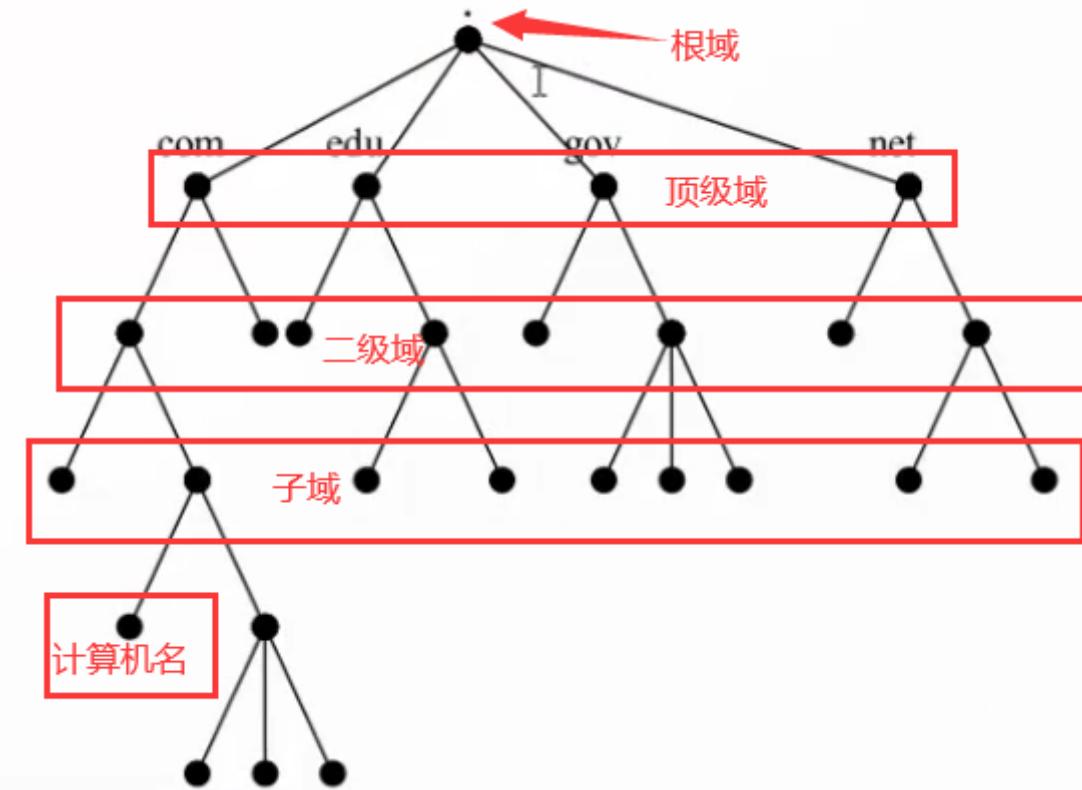
一、hosts文件：(解析先从本地开始，本地没有才会查询DNS服务器)

作用：事先名字解析，主要为本地主机名解析，集群节点提供快速解析

数据库：平面式结构，集中式数据库

二、DNS域名服务

作用：将域名解析成IP //（ARP将IP转换为MAC）



惟缓存dns：cache only

装dns的软件包：yum -y bind bind-chroot

dns主配置文件：/etc/named.conf

刚开始配置 options里：listen-on port 53 {any}; //或者把这行删除掉 listen-on-v6 port 53{any}; //或者把这行删掉 any是所有人都能访问 allow-query{any};//或删掉 //允许的访问

打开服务：systemctl start named

配置防火墙

根(.)提示

dns转发forward[通常转发到上一级的DNS服务器]

options{

.....

forwarders{114.114.114.114;202.106.0.20;}//在option中添加forward转发 //写当地的ISP给提供的DNS服务器

}

客户端指定dns路径：vim /etc/resolv.conf //nameserver

dig命令 用 yum -y install bind-utils

正向区解析设置：

正向区：提供正向解析，将域名解析为IP。

反向区：提供反向解析，将IP解析为域名。

一个dns可以管理多个域。

两步：1.在dns主配置文件里添加新域(zone)；2.添加数据库文件内容

1.主配置文件 相当于在域名服务商申请域名

```
vim /etc/named.conf
```

```
options{
    directory "/var/named"; //数据库文件存储位置
};

zone "wo.com"{
    type master; //创建域wo.com
    file "wo.com.zone"; //file为相对路径相对于上边的directory
};
```

2.数据库文件(区域文件) 在域名服务提供的页面上做解析

```
@//当前域名wo.com.          IN      SOA //起始授权      dns //授权给的主机(完整
为:dns.wo.com.)           root //出现问题时发给哪个邮箱(完整为root.wo.com.)   (2020120200
1H 10M 1W 1D) //版本号 辅助dns同步时间 尝试同步时间 放弃同步时间 缓存时间
@      IN      NS      dns
dns    IN      A       192.168.241.128
www    IN      A       192.168.241.128 //www为完全限定域名
@      IN      A       193.168.241.128 //将本地的http服务解析
如果单独是个域名直接写@代替当前域名，如果后面还得加域名可以省略，例: dns.wo.com.=dns
```

```
@      IN      SOA      dns      root      (2020120200 1H 10M 1W 1D)
@      IN      NS      dns
dns    IN      A       192.168.241.128
www    IN      A       192.168.241.128
@      IN      A       193.168.241.128
~
```

版本号(自己随意定义,10位以内都行,一般以时间为例子年月日再添加两位修正位) 1H 15M 1W 1D //辅助dns同步时间 尝试同步时间 放弃同步时间 缓存时间 //每隔1小时辅助dns会来统计一次,如果辅助dns发现能与主dns同步成功,会在下一个一小时再来同步;若没同步成功则在15分钟后来重试,若还是不行会每隔15分一直重试,当重试时间达到一周还没同步成功则放弃同步

@ 表示当前域名

[www.wo.com.](#) = www

SOA:起始授权记录 强制(必须有)

NS:DNS服务器记录 强制

A:主机记录

CNAME:别名记录

```
vim /var/named/wo.com.zone
```

