

Linux 系统目录结构

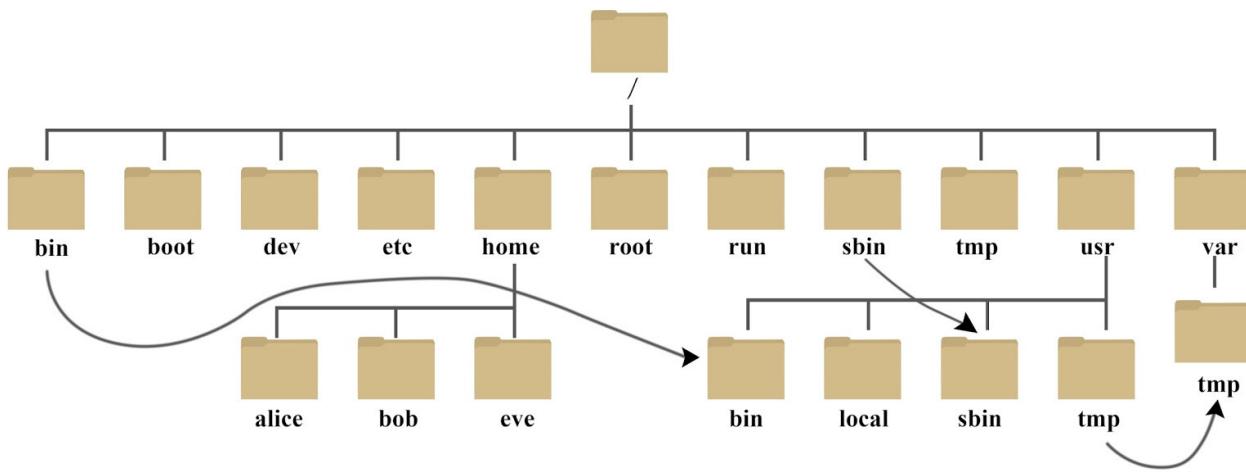
登录系统后，在当前命令窗口下输入命令：

```
ls /
```

你会看到如下图所示：

```
[root@localhost ~]# ls /
bin  dev  home  lost+found  mnt  proc  sbin  srv  tmp  var
boot  etc  lib   media      opt  root  selinux  sys  usr
```

树状目录结构：



以下是对这些目录的解释：

- **/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, 如上图中的 alice、bob 和 eve。

- **/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：

```
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 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外的普通用户），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。

在 Linux 或 Unix 操作系统中，所有的文件和目录都被组织成以一个根节点开始的倒置的树状结构。

文件系统的最顶层是由根目录开始的，系统使用 / 来表示根目录。在根目录之下的既可以是目录，也可以是文件，而每一个目录中又可以包含子目录文件。如此反复就可以构成一个庞大的文件系统。

在Linux文件系统中有两个特殊的目录，一个用户所在的工作目录，也叫当前目录，可以使用一个点`.`来表示；另一个是当前目录的上一级目录，也叫父目录，可以使用两个点`..`来表示。

- `.`：代表当前的目录，也可以使用`./`来表示；
- `..`：代表上一层目录，也可以`../`来代表。

如果一个目录或文件名以一个点`.`开始，表示这个目录或文件是一个隐藏目录或文件(如：`.bashrc`)。即以默认方式查找时，不显示该目录或文件。

系统启动必须：

- `/boot`：存放的启动Linux时使用的内核文件，包括连接文件以及镜像文件。
- `/etc`：存放所有的系统需要的配置文件和子目录列表，更改目录下的文件可能会导致系统不能启动。
- `/lib`：存放基本代码库（比如c++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。
- `/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

指令集合：

- `/bin`：存放着最常用的程序和指令
- `/sbin`：只有系统管理员能使用的程序和指令。

外部文件管理：

- `/dev`：Device(设备)的缩写，存放的是Linux的外部设备。注意：在Linux中访问设备和访问文件的方式是相同的。
- `/media`：类windows的其他设备，例如U盘、光驱等等，识别后linux会把设备放到这个目录下。
- `/mnt`：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。

临时文件：

- `/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`。
- `/lost+found`：一般情况下为空的，系统非法关机后，这里就存放一些文件。
- `/tmp`：这个目录是用来存放一些临时文件的。

账户：

- **/root**: 系统管理员的用户主目录。
- **/home**: 用户的主目录，以用户的账号命名的。
- **/usr**: 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program files目录。
- **/usr/bin**: 系统用户使用的应用程序与指令。
- **/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。

运行过程中要用：

- **/var**: 存放经常修改的数据，比如程序运行的日志文件（/var/log 目录下）。
- **/proc**: 管理内存空间！虚拟的目录，是系统内存的映射，我们可以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来做修改。

扩展用的：

- **/opt**: 默认是空的，我们安装额外软件可以放在这个里面。
 - **/srv**: 存放服务启动后需要提取的数据（不用服务器就是空）
-

Linux 文件与目录管理

我们知道Linux的目录结构为树状结构，最顶级的目录为根目录 /。

其他目录通过挂载可以将它们添加到树中，通过解除挂载可以移除它们。

在开始本教程前我们需要先知道什么是绝对路径与相对路径。

• 绝对路径：

路径的写法，由根目录 / 写起，例如： /usr/share/doc 这个目录。

•

相对路径：

路径的写法，不是由 / 写起，例如由 /usr/share/doc 要到 /usr/share/man 底下时，可以写成： cd .. / man
这就是相对路径的写法。

处理目录的常用命令

接下来我们就来看几个常见的处理目录的命令吧：

- **ls** (英文全拼： list files) : 列出目录及文件名
- **cd** (英文全拼： change directory) : 切换目录

- `pwd` (英文全拼: print work directory) : 显示目前的目录
- `mkdir` (英文全拼: make directory) : 创建一个新的目录
- `rmdir` (英文全拼: remove directory) : 删除一个空的目录
- `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
- `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

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

ls (列出目录)

在Linux系统当中, `ls` 命令可能是最常被运行的。

语法:

```
[root@www ~]# ls [-aAdfFhilnrRSt] 目录名称  
[root@www ~]# ls [--color={never,auto,always}] 目录名称  
[root@www ~]# ls [--full-time] 目录名称
```

选项与参数:

- `-a` : 全部的文件, 连同隐藏文件(开头为`.`的文件)一起列出来(常用)
- `-d` : 仅列出目录本身, 而不是列出目录内的文件数据(常用)
- `-l` : 长数据串列出, 包含文件的属性与权限等等数据; (常用)

将家目录下的所有文件列出来(含属性与隐藏档)

```
[root@www ~]# ls -al ~
```

cd (切换目录)

`cd`是`Change Directory`的缩写, 这是用来变换工作目录的命令。

语法:

```
cd [相对路径或绝对路径]  
#使用 mkdir 命令创建 runoob 目录  
[root@www ~]# mkdir runoob  
  
#使用绝对路径切换到 runoob 目录  
[root@www ~]# cd /root/runoob/  
  
#使用相对路径切换到 runoob 目录  
[root@www ~]# cd ./runoob/  
  
# 表示回到自己的家目录, 亦即是 /root 这个目录  
[root@www runoob]# cd ~  
  
# 表示去到目前的上一级目录, 亦即是 /root 的上一级目录的意思;  
[root@www ~]# cd ..
```

接下来大家多操作几次应该就可以很好的理解 cd 命令的。

pwd (显示目前所在的目录)

pwd 是 Print Working Directory 的缩写，也就是显示目前所在目录的命令。

```
[root@www ~]# pwd [-P]
```

选项与参数：

- -P：显示出确实的路径，而非使用连结 (link) 路径。

实例：单纯显示出目前的工作目录：

```
[root@www ~]# pwd  
/root <== 显示出目录啦 ~
```

实例显示出实际的工作目录，而非连结档本身的目录名而已。

```
[root@www ~]# cd /var/mail <==注意，/var/mail是一个连结档  
[root@www mail]# pwd  
/var/mail <==列出目前的工作目录  
[root@www mail]# pwd -P  
/var/spool/mail <==怎么回事？有没有加 -P 差很多～  
[root@www mail]# ls -ld /var/mail  
lrwxrwxrwx 1 root root 10 Sep 4 17:54 /var/mail -> spool/mail  
# 看到这里应该知道为啥了吧？因为 /var/mail 是连结档，连结到 /var/spool/mail  
# 所以，加上 pwd -P 的选项后，会不以连结档的数据显示，而是显示正确的完整路径啊！
```

mkdir (创建新目录)

如果想要创建新的目录的话，那么就使用mkdir (make directory)吧。

语法：

```
mkdir [-mp] 目录名称
```

选项与参数：

- -m：配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～
- -p：帮助你直接将所需要的目录(包含上一级目录)递归创建起来！

实例：请到/tmp底下尝试创建数个新目录看看：

```
[root@www ~]# cd /tmp  
[root@www tmp]# mkdir test <==创建一名为 test 的新目录  
[root@www tmp]# mkdir test1/test2/test3/test4  
mkdir: cannot create directory `test1/test2/test3/test4':  
No such file or directory <== 没办法直接创建此目录啊！  
[root@www tmp]# mkdir -p test1/test2/test3/test4
```

加了这个 -p 的选项，可以自行帮你创建多层目录！

实例：创建权限为 rwx--x--x 的目录。

```
[root@www tmp]# mkdir -m 711 test2
[root@www tmp]# ls -l
drwxr-xr-x 3 root root 4096 Jul 18 12:50 test
drwxr-xr-x 3 root root 4096 Jul 18 12:53 test1
drwx--x--x 2 root root 4096 Jul 18 12:54 test2
```

上面的权限部分，如果没有加上 -m 来强制配置属性，系统会使用默认属性。

如果我们使用 -m，如上例我们给予 -m 711 来给予新的目录 drwx--x--x 的权限。

rmmdir (删除空的目录)

语法：

```
rmmdir [-p] 目录名称
```

选项与参数：

- -p：连同上一级『空的』目录也一起删除

删除 runoob 目录

```
[root@www tmp]# rmmdir runoob/
```

将 mkdir 实例中创建的目录(/tmp 底下)删除掉！

```
[root@www tmp]# ls -l  <==看看有多少目录存在?
drwxr-xr-x 3 root root 4096 Jul 18 12:50 test
drwxr-xr-x 3 root root 4096 Jul 18 12:53 test1
drwx--x--x 2 root root 4096 Jul 18 12:54 test2
[root@www tmp]# rmmdir test  <==可直接删除掉，没问题
[root@www tmp]# rmmdir test1 <==因为尚有内容，所以无法删除!
rmmdir: `test1': Directory not empty
[root@www tmp]# rmmdir -p test1/test2/test3/test4
[root@www tmp]# ls -l          <==您看看，底下的输出中test与test1不见了!
drwx--x--x 2 root root 4096 Jul 18 12:54 test2
```

利用 -p 这个选项，立刻就可以将 test1/test2/test3/test4 一次删除。

不过要注意的是，这个 rmmdir 仅能删除空的目录，你可以使用 rm 命令来删除非空目录。

cp (复制文件或目录)

cp 即拷贝文件和目录。

语法：

```
[root@www ~]# cp [-adfilprs] 来源档(source) 目标档(destination)
[root@www ~]# cp [options] source1 source2 source3 ... directory
```

选项与参数：

- **-a**: 相当於 -pd़r 的意思，至於 pd़r 请参考下列说明；(常用)
- **-d**: 若来源档为连结档的属性(link file)，则复制连结档属性而非文件本身；
- **-f**: 为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次；
- **-i**: 若目标档(destination)已经存在时，在覆盖时会先询问动作的进行(常用)
- **-l**: 进行硬式连结(hard link)的连结档创建，而非复制文件本身；
- **-p**: 连同文件的属性一起复制过去，而非使用默认属性(备份常用)；
- **-r**: 递归持续复制，用於目录的复制行为；(常用)
- **-s**: 复制成为符号连结档 (symbolic link)，亦即『捷径』文件；
- **-u**: 若 destination 比 source 旧才升级 destination !

用 root 身份，将 root 目录下的 .bashrc 复制到 /tmp 下，并命名为 bashrc

```
[root@www ~]# cp ~/.bashrc /tmp/bashrc
[root@www ~]# cp -i ~/.bashrc /tmp/bashrc
cp: overwrite `/tmp/bashrc'? n <==n不覆盖, y为覆盖
```

rm (移除文件或目录)

语法：

```
rm [-fir] 文件或目录
```

选项与参数：

- **-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**: 递归删除啊！最常用在目录的删除了！这是非常危险的选项！！！
-

将刚刚在 cp 的实例中创建的 bashrc 删除掉！

```
[root@www tmp]# rm -i bashrc
rm: remove regular file `bashrc'? y
```

如果加上 -i 的选项就会主动询问喔，避免你删除到错误的档名！

mv (移动文件与目录，或修改名称)

语法：

```
[root@www ~]# mv [-fiu] source destination
[root@www ~]# mv [options] source1 source2 source3 .... directory
```

选项与参数：

- **-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)

复制一文件，创建一目录，将文件移动到目录中

```
[root@www ~]# cd /tmp  
[root@www tmp]# cp ~/.bashrc bashrc  
[root@www tmp]# mkdir mvtest  
[root@www tmp]# mv bashrc mvtest
```

将某个文件移动到某个目录去，就是这样做！

将刚刚的目录名称更名为 mvtest2

```
[root@www tmp]# mv mvtest mvtest2
```

Linux 文件内容查看

Linux系统中使用以下命令来查看文件的内容：

- **cat** 由第一行开始显示文件内容
- **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！
- **more** 一页一页的显示文件内容
- **less** 与 more 类似，但是比 more 更好的是，他可以往前翻页！
- **head** 只看头几行
- **tail** 只看尾巴几行

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

cat

由第一行开始显示文件内容

语法：

```
cat [-AbEnTv]
```

选项与参数：

- **-A**：相当於 -vET 的整合选项，可列出一些特殊字符而不是空白而已；
- **-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 ^I 显示出来；
- -v：列出一些看不出来的特殊字符

检看 /etc/issue 这个文件的内容：

```
[root@www ~]# cat /etc/issue
CentOS release 6.4 (Final)
Kernel \r on an \m
```

tac

tac与cat命令刚好相反，文件内容从最后一行开始显示，可以看出 tac 是 cat 的倒着写！如：

```
[root@www ~]# tac /etc/issue

Kernel \r on an \m
CentOS release 6.4 (Final)
```

nl

显示行号

语法：

```
nl [-bnw] 文件
```

选项与参数：

- -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。

实例一：用 nl 列出 /etc/issue 的内容

```
[root@www ~]# nl /etc/issue
1  CentOS release 6.4 (Final)
2  Kernel \r on an \m
```

more

一页一页翻动

```
[root@www ~]# more /etc/man_db.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
....(中间省略)....
--More--(28%) <== 重点在这一行喔！你的光标也会在这里等待你的命令
```

在 more 这个程序的运行过程中，你有几个按键可以按的：

- 空白键 (space)：代表向下翻一页；
- Enter : 代表向下翻『一行』；
- /字串 : 代表在这个显示的内容当中，向下搜寻『字串』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

一页一页翻动，以下实例输出/etc/man.config文件的内容：

```
[root@www ~]# less /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
....(中间省略)....
: <== 这里可以等待你输入命令！
```

less运行时可以输入的命令有：

- 空白键 : 向下翻动一页；
- [pagedown]: 向下翻动一页；
- [pageup]: 向上翻动一页；
- /字串 : 向下搜寻『字串』的功能；
- ?字串 : 向上搜寻『字串』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

取出文件前面几行

语法：

```
head [-n number] 文件
```

选项与参数：

- -n：后面接数字，代表显示几行的意思

```
[root@www ~]# head /etc/man.config
```

默认的情况下，显示前面 10 行！若要显示前 20 行，就得要这样：

```
[root@www ~]# head -n 20 /etc/man.config
```

tail

取出文件后面几行

语法：

```
tail [-n number] 文件
```

选项与参数：

- -n：后面接数字，代表显示几行的意思
- -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

```
[root@www ~]# tail /etc/man.config  
# 默认的情况下，显示最后的十行！若要显示最后的 20 行，就得要这样：  
[root@www ~]# tail -n 20 /etc/man.config
```

软连接（连接类型）

命令：`ln -s 原文件 链接路径`

```
[root@localhost ghTest]# ln -s a.txt /ghTest/111  
[root@localhost ghTest]# ll  
total 8  
lrwxrwxrwx. 1 root root 5 Dec 1 12:38 111 -> a.txt  
-rw-r--r--. 1 root root 35 Nov 29 09:00 a.txt
```

软连接记录的只是文件的绝对路径，失去原文件不可用

硬链接

硬链接不依赖于原文件，失去原文件依旧可以，但是不可以对目录做硬链接

只能在同一个分区做硬链接

Linux vi/vim

所有的 Unix Like 系统都会内建 vi 文书编辑器，其他的文书编辑器则不一定会存在。

但是目前我们使用比较多的是 vim 编辑器。

vim 具有程序编辑的能力，可以主动的以字体颜色辨别语法的正确性，方便程序设计。

什么是 vim?

Vim是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。

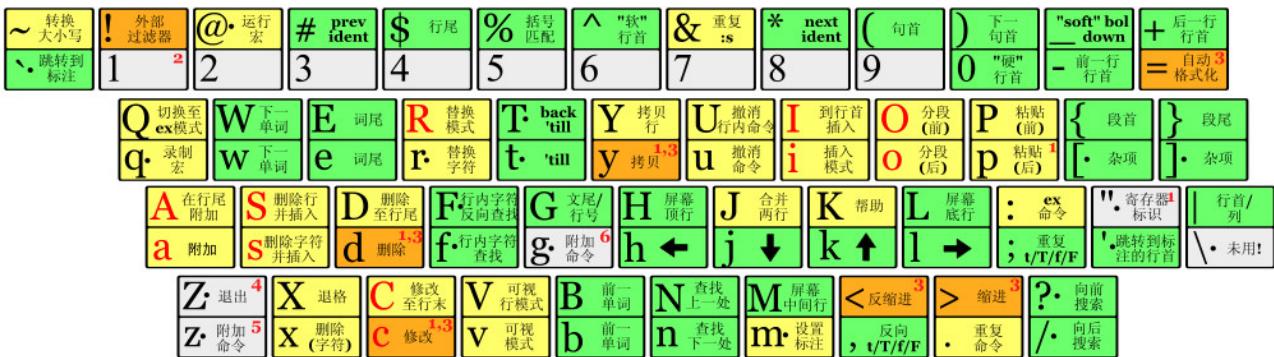
简单的来说， vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。 vim 则可以说是程序开发者的一项很好用的工具。

连 vim 的官方网站 (<http://www.vim.org>) 自己也说 vim 是一个程序开发工具而不是文字处理软件。

vim 键盘图：

vi / vim 键盘图

Esc
命令模式



动作 移动光标, 或者定义操作的范围

命令 直接执行的命令, 红色命令进入编辑模式

操作 后面跟随表示操作范围的指令

extra 特殊功能, 需要额外的输入

Q· 后跟字符参数

w,e,b命令

小写(b): `guux(foo, bar, baz);`
大写(B): `guux(foo, bar, baz);`

主要ex命令:

:w (保存), :q (退出), :q! (不保存退出)
:e f (打开文件 f),
:os/x/y/g ('y' 全局替换 'x'),
:h (帮助 in vim), :new (新建文件 in vim),

其它重要命令:

CTRL-R: 重复 (vim),
CTRL-F/-B: 上翻/下翻,
CTRL-E/-Y: 上滚/下滚,
CTRL-V: 块可视模式 (vim only)

可视模式:

漫游后对选中的区域执行操作 (vim only)

备注:

(1) 在拷贝/粘贴/删除 命令前使用 "x (x=a..z,*)

使用命令的寄存器('剪贴板')
(如: "ay\$ 拷贝剩余的行内容至寄存器 'a')

(2) 命令前添加数字

多遍重复操作

(e.g.: 2p, d2w, 5i, d4j)

(3) 重复本字符在光标所在行执行操作

(dd = 删除本行, >> = 行首缩进)

(4) ZZ 保存退出, ZQ 不保存退出

(5) zt: 移动光标所在行至屏幕顶端,

zb: 底端, zz: 中间

(6) gg: 行首 (vim only),

gf: 打开光标处的文件名 (vim only)

原图: www.viemu.com 翻译: fdl (linuxsir)

vi/vim 的使用

基本上 vi/vim 共分为三种模式, 分别是**命令模式 (Command mode)**, **输入模式 (Insert mode)** 和**底线命令模式 (Last line mode)**。这三种模式的作用分别是:

命令模式

用户刚刚启动 vi/vim, 便进入了命令模式。

此状态下敲击键盘动作会被Vim识别为命令, 而非输入字符。比如我们此时按下i, 并不会输入一个字符, i被当作了一个命令。

以下是常用的几个命令:

- i 切换到输入模式, 以输入字符。
- 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。

若想要编辑文本: 启动Vim, 进入了命令模式, 按下i, 切换到输入模式。

命令模式只有一些最基本的命令, 因此仍要依靠底线命令模式输入更多命令。

输入模式

在命令模式下按下i就进入了输入模式。

在输入模式中，可以使用以下按键：

- **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

底线命令模式

在命令模式下按下:（英文冒号）就进入了底线命令模式。

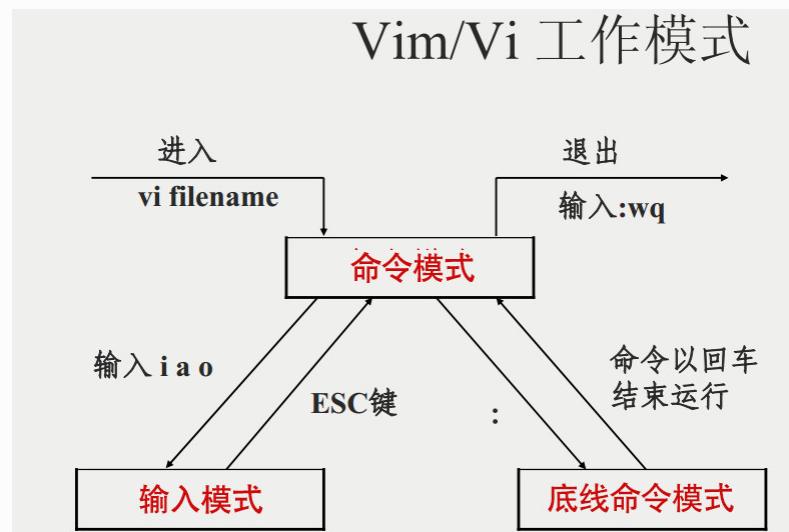
底线命令模式可以输入单个或多个字符的命令，可用的命令非常多。

在底线命令模式中，基本的命令有（已经省略了冒号）：

- q 退出程序
- w 保存文件

按ESC键可随时退出底线命令模式。

简单的说，我们可以将这三个模式想成底下的图标来表示：



vi/vim 使用实例

使用 vi/vim 进入一般模式

如果你想要使用 vi 来建立一个名为 runoob.txt 的文件时，你可以这样做：

```
$ vim runoob.txt
```

直接输入 **vi 文件名** 就能够进入 vi 的一般模式了。请注意，记得 vi 后面一定要加文件名，不管该文件存在与否！



按下 i 进入输入模式(也称为编辑模式)

在一般模式之中，只要按下 i, o, a 等字符就可以进入输入模式了！

在编辑模式当中，你可以发现在左下角状态栏中会出现 -INSERT- 的字样，那就是可以输入任意字符的提示。

这个时候，键盘上除了 Esc 这个按键之外，其他的按键都可以视作为一般的输入按钮了，所以你可以进行任何的编辑。



按下 ESC 按钮回到一般模式

好了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！没错！就是给他按下 Esc 这个按钮即可！马上你就会发现画面左下角的 - INSERT - 不见了！

在一般模式中按下 :wq 储存后离开 vi

OK，我们要存档了，存盘并离开的指令很简单，输入 :wq 即可保存离开！



OK! 这样我们就成功创建了一个 runoob.txt 的文件。

vi/vim 按键说明

除了上面简易范例的 i, Esc, :wq 之外，其实 vim 还有非常多的按键可以使用。

第一部分：一般模式可用的光标移动、复制粘贴、搜索替换等

移动光标的方法	
h 或 向左箭头键(←)	光标向左移动一个字符
j 或 向下箭头键(↓)	光标向下移动一个字符
k 或 向上箭头键(↑)	光标向上移动一个字符
l 或 向右箭头键(→)	光标向右移动一个字符
如果你将右手放在键盘上的话，你会发现 hjkl 是排列在一起的，因此可以使用这四个按钮来移动光标。如果想要进行多次移动的话，例如向下移动 30 行，可以使用 "30j" 或 "30↓" 的组合按键，亦即加上想要进行的次数(数字)后，按下动作即可！	
[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	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 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	n 为数字。光标向下移动 n 行(常用)
搜索替换	
/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。

移动光标的方法	
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	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)
: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)
删除、复制与贴上	
x, X	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	删除游标所在的那一整行(常用)
ndd	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
yy	复制游标所在那一行(常用)
nyy	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	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	复原前一个动作。(常用)
[Ctrl]+r	重做上一个动作。(常用)

移动光标的方法	
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)

第二部分：一般模式切换到编辑模式的可用的按钮说明

进入输入或取代的编辑模式	
i, I	进入输入模式(Insert mode): i 为『从目前光标所在处输入』， I 为『在目前所在行的第一个非空格符处开始输入』。(常用)
a, A	进入输入模式(Insert mode): a 为『从目前光标所在的下一个字符处开始输入』， A 为『从光标所在行的最后一个字符处开始输入』。(常用)
o, O	进入输入模式(Insert mode): 这是英文字母 o 的大小写。o 为『在目前光标所在的下一行处输入新的一行』； O 为在目前光标所在处的上一行输入新的一行！(常用)
r, R	进入取代模式(Replace mode): r 只会取代光标所在的那一个字符一次； R 会一直取代光标所在的文字，直到按下 ESC 为止；(常用)
上面这些按键中，在 vi 画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样。由名称就知道该动作了吧！！特别注意的是，我们上面也提过了，你想要在档案里面输入字符时，一定要在左下角处看到 INSERT 或 REPLACE 才能输入喔！	
[Esc]	退出编辑模式，回到一般模式中(常用)

第三部分：一般模式切换到指令行模式的可用的按钮说明

指令行的储存、离开等指令	
:w	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！
: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 喔！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!。
:w [filename]	将编辑的数据储存成另一个档案 (类似另存新档)
:r [filename]	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
: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！
vim 环境的变更	
:set nu	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！

特别注意，在 vi/vim 中，数字是很有意义的！数字通常代表重复做几次的意思！

也有可能是代表去到第几个什么什么的意思。

举例来说，要删除 50 行，则是用『50dd』，数字加在动作之前，向下移动 20 行呢？那就是『20j』或者是『20↓』

批量操作

批量注释：

Ctrl + v 进入块选择模式，然后移动光标选中你要注释的行，再按大写的 **I** 进入行首插入模式输入注释符号如 // 或 #，输入完毕之后，按两下 **ESC**，Vim 会自动将你选中的所有行首都加上注释，保存退出完成注释。

取消注释：

Ctrl + v 进入块选择模式，选中你要删除的行首的注释符号，注意 // 要选中两个，选好之后按 **d** 即可删除注释，**ESC** 保存退出。

方法二：替换命令

批量注释。

使用下面命令在指定的行首添加注释。

使用名命令格式： :起始行号,结束行号s/^/注释符/g (注意冒号) 。

取消注释：

使用名命令格式： :起始行号,结束行号s/^注释符//g (注意冒号) 。

例子：

1、在 10 - 20 行添加 // 注释

```
:10,20s/^##//g
```

2、在 10 - 20 行删除 // 注释

```
:10,20s/^##//g
```

3、在 10 - 20 行添加 # 注释

```
:10,20s/^#/g
```

4、在 10 - 20 行删除 # 注释

```
:10,20s/#//g
```

Linux用户管理

三个主要文件

Linux 系统中用户信息存放在 /etc/passwd 文件中

这是一个包含每个用户基本信息的文本文件。当我们在系统中创建一个用户，新用户的详细信息就会被添加到这个文件中

/etc/passwd 内容解释

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

用户名、密码

已创建用户的用户名，字符长度 1 个到 12 个字符，“x”代表加密密码保存在 /etc/shadow 文件 中

UID(身份证号)

代表用户的 ID 号，每个用户都要有一个唯一的 ID

UID 号为 0 的是为 root 用户保留的

UID 号 1 到 99 是为系统用户保留的

UID 号 100-999 是为系统账户和群组保留的

GID (身份组号)

代表群组的 ID 号，每个群组都要有一个唯一的 GID

保存在 [/etc/group文件](#) 中

root (身份描述)

描述用户的信息

此用户登陆系统时所在的家目录

登录shell (命令解释器)

/etc/shadow 内容解释

```
root:$6$P4ITD/PL/o9.8tmm$kTTL7t0IIDoJ4X5spyDlqAsg17UAuPaD1MhKAScCjkkwGYH9mSbAGa37LgdE1qPP
okPaZmg3b49jajhjT6FUS1::0:99999:7:::
```

1) “登录名” 是与 /etc/passwd 文件中的登录名相一致的用户账号
2) “口令” 字段存放的是加密后的用户口令字，如果为空，则对应用户没有口令，登录时不需要口令；
星号代表帐号被锁定；
双叹号表示这个密码已经过期了。
\$6\$开头的，表明是用SHA-512加密的，
\$1\$ 表明是用MD5加密的
\$2\$ 是用Blowfish加密的
\$5\$ 是用 SHA-256加密的。

aaa:\$6\$DBd:18302:0:99999:7: :	:1							

1	2	3	4	5	6	7	8	9
1. 用户名								
2. 密码加密值								
3. 最后一次修改时间，2020年2月10，								过了多少天？
4. 最小间隔（0代表当天可以改密码）								
5. 密码最大时间间隔，9999无限期								
6. 警告时间（7）								
7. 不活动时间（28，用户不登录系统。超过28天，禁用你。）								
8. 失效时间								
9. 保留								

/etc/group

```
root:x:0:
```

组名、组密码、组ID、组成员

添加用户

```
useradd 选项 用户名
```

参数说明：

- 选项：

- -c comment 指定一段注释性描述
- -d 目录 指定用户主目录，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 用户组，用户组 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 指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号

- 用户名：

指定新账号的登录名

实例1：

```
# useradd -d /home/sam -m sam
```

此命令创建了一个用户sam，其中-d和-m选项用来为登录名sam产生一个主目录 /home/sam

(/home为默认的用户主目录所在的父目录)

实例2：

```
# useradd -s /bin/sh -g group -G adm,root gem
```

此命令新建了一个用户gem

该用户的登录Shell是 **/bin/sh**

它属于group用户组，同时又属于adm和root用户组，其中group用户组是其主组

这里可能新建组：**#groupadd group及groupadd adm**

增加用户账号就是在/etc/passwd文件中为新用户增加一条记录，同时更新其他系统文件如/etc/shadow, /etc/group等

Linux提供了集成的**系统管理工具userconf**，它可以用来对用户账号进行统一管理

删除用户

如果一个用户的账号不再使用，可以从系统中删除。删除用户账号就是要将/etc/passwd等系统文件中的该用户记录删除，必要时还删除用户的主目录

删除一个已有的用户账号使用 `userdel` 命令，其格式如下：

```
userdel 选项 用户名
```

常用的选项是 `-r`，它的作用是把用户的主目录一起删除

例如：

```
# userdel -r sam
```

此命令删除用户sam在系统文件中（主要是/etc/passwd, /etc/shadow, /etc/group等）的记录，同时删除用户的主目录

修改帐号相关信息

修改用户账号就是根据实际情况更改用户的有关属性，如用户号、主目录、用户组、登录Shell等

修改已有用户的信息使用 `usermod` 命令，其格式如下：

```
usermod 选项 用户名
```

常用的选项包括 `-c, -d, -m, -g, -G, -s, -u` 以及 `-o` 等，这些选项的意义与 `useradd` 命令中的选项一样，可以为用户指定新的资源值

另外，有些系统可以使用选项：`-l 新用户名`

这个选项指定一个新的账号，即将原来的用户名改为新的用户名

例如：

```
# usermod -s /bin/ksh -d /home/z -g developer sam
```

此命令将用户sam的登录Shell修改为ksh，主目录改为/home/z，用户组改为developer。

用户密码的管理

用户管理的一项重要内容是用户密码的管理。用户账号刚创建时没有密码，但是被系统锁定，无法使用，必须为其指定密码后才可以使用，即使是指定空密码。

指定和修改用户密码的Shell命令是 `passwd`。超级用户可以为自己和其他用户指定密码，普通用户只能用它修改自己的密码。命令的格式为：

```
passwd 选项 用户名
```

可使用的选项：

- -l 锁定密码，即禁用账号
- -u 密码解锁
- -d 使账号无密码
- -f 强迫用户下次登录时修改密码

如果默认用户名，则修改当前用户的密码

例如，假设当前用户是sam，则下面的命令修改该用户自己的密码：

```
$ passwd  
Old password:*****  
New password:*****  
Re-enter new password:*****
```

如果是超级用户，可以用下列形式指定任何用户的密码：

```
# passwd sam  
New password:*****  
Re-enter new password:*****
```

普通用户修改自己的密码时，passwd命令会先询问原密码，验证后再要求用户输入两遍新密码，如果两次输入的密码一致，则将这个密码指定给用户；而超级用户为用户指定密码时，就不需要知道原密码

为了系统安全起见，用户应该选择比较复杂的密码，例如最好使用8位长的密码，密码中包含有大写、小写字母和数字，并且应该与姓名、生日等不相同

为用户指定空密码时，执行下列形式的命令：

```
# passwd -d sam
```

此命令将用户 sam 的密码删除，这样用户 sam 下一次登录时，系统就不再允许该用户登录了

passwd 命令还可以用 -l(lock) 选项锁定某一用户，使其不能登录，例如：

```
# passwd -l sam
```

添加批量用户

添加和删除用户对每位Linux系统管理员都是轻而易举的事，比较棘手的是如果要添加几十个、上百个甚至上千个用户时，我们不太可能还使用useradd一个一个地添加，必然要找一种简便的创建大量用户的方法。Linux系统提供了创建大量用户的工具，可以让您立即创建大量用户，方法如下：

(1) 先编辑一个文本用户文件。

每一列按照 `/etc/passwd` 密码文件的格式书写，要注意每个用户的用户名、UID、宿主目录都不可以相同，其中密码栏可以留做空白或输入x号。一个范例文件`user.txt`内容如下：

```
user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::603:100:user:/home/user004:/bin/bash
user005::604:100:user:/home/user005:/bin/bash
user006::605:100:user:/home/user006:/bin/bash
```

(2) 以root身份执行命令 `/usr/sbin/newusers`，从刚创建的用户文件 `user.txt` 中导入数据，创建用户：

```
# newusers < user.txt
```

然后可以执行命令 `vipw` 或 `vi /etc/passwd` 检查 `/etc/passwd` 文件是否已经出现这些用户的数据，并且用户的宿主目录是否已经创建

(3) 执行命令`/usr/sbin/pwunconv`。

将 `/etc/shadow` 产生的 `shadow` 密码解码，然后回写到 `/etc/passwd` 中，并将 `/etc/shadow` 的 `shadow` 密码栏删掉。这是为了方便下一步的密码转换工作，即先取消 `shadow password` 功能

```
# pwunconv
```

(4) 编辑每个用户的密码对照文件。

格式为：

用户名:密码

实例文件 `passwd.txt` 内容如下：

```
user001:123456
user002:123456
user003:123456
user004:123456
user005:123456
user006:123456
```

(5) 以 root 身份执行命令 `/usr/sbin/chpasswd`。

创建用户密码，`chpasswd` 会将经过 `/usr/bin/passwd` 命令编码过的密码写入 `/etc/passwd` 的密码栏

```
# chpasswd < passwd.txt
```

(6) 确定密码经编码写入/etc/passwd的密码栏后。

执行命令 `/usr/sbin/pwconv` 将密码编码为 `shadow password`，并将结果写入 `/etc/shadow`

```
# pwconv
```

Linux用户组管理

每个用户都有一个用户组，系统可以对一个用户组中的所有用户进行集中管理。

基本组：不同Linux系统对用户组的规定有所不同，如 [Linux下的用户属于与它同名的用户组](#)，这个用户组在创建用户时同时创建

用户组的管理涉及用户组的添加、删除和修改

组的增加、删除和修改实际上就是对/etc/group文件的更新

增加用户组

```
groupadd 选项 用户组
```

可以使用的选项有：

- `-g GID` 指定新用户组的组标识号（GID）
- `-o` 一般与`-g`选项同时使用，表示新用户组的GID可以与系统已有用户组的GID相同

实例1：

```
# groupadd group1
```

此命令向系统中增加了一个新组group1，新组的组标识号是在当前已有的最大组标识号的基础上加1

实例2：

```
# groupadd -g 101 group2
```

此命令向系统中增加了一个新组group2，同时指定新组的组标识号是101

删除已有的用户组

```
groupdel 用户组
```

例如：

```
# groupdel group1
```

此命令从系统中删除组group1

修改用户组的属性

```
groupmod 选项 用户组
```

常用的选项有：

- -g GID 为用户组指定新的组标识号
- -o 与-g选项同时使用，用户组的新GID可以与系统已有用户组的GID相同
- -n新用户组 将用户组的名字改为新名字

实例1

```
# groupmod -g 102 group2
```

此命令将组group2的组标识号修改为102

实例2

```
# groupmod -g 10000 -n group3 group2
```

此命令将组group2的标识号改为10000，组名修改为group3

用户组之间切换

用户可以在登录后，使用命令newgrp切换到其他用户组，这个命令的参数就是目的用户组。例如：

```
$ newgrp root
```

这条命令将当前用户切换到root用户组，前提条件是root用户组确实是该用户的主组或附加组。类似于用户账号的管理，用户组的管理也可以通过集成的系统管理工具来完成

Linux用户切换

一般用户切换是为了获得权限，有两种方式

永久提权

高级用户往下不用输密码，反之需要

此时身份和root完全相同（就是登陆了root）且只要不关机，不退出，一直有效

```
SU - root  
SU 用户名
```

暂时提权

只有部分高级指令可以使用,指令权限怎么分配的呢?

授权 (root授权)

```
/etc/sudoers
```

在这个文件中修改相关信息

使用

```
sudo useradd 用户名
```

Linux用户权限

权限对象

u	属主
g	属组
o	其他人
所有人	a(u + g + o)

权限类型

读	r	4
写	w	2
执行	x	1

文件II详细展示信息

```
- rw- r-- r--. 1 root root 35 Nov 29 09:00 a.txt  
文件类型 属主权限 属组 其他人 链接数 属主 属组 文件大小 创建时间 文件名
```

基本权限UGO

设置权限

有两种形式，字母和数字是对应的关系

chmod命令

- -R 对文件夹下所有文件进行权限修改，否则只是文件夹权限修改

使用字母

chmod 选项 对象(u/g/o)赋值符(+/-/=)权限类型 (r/w/x) 目标文件、目录

```
chmod u+r 1.txt  
chmod u-r 1.txt  
chmod u=wr 1.txt
```

使用数字

chmod 选项 数字 目标文件、目录

```
chmod 760 1.txt  
chmod 710 1.txt  
chmod 777 1.txt
```

更改属主、属组

chown命令

- -R对文件夹下所有文件进行权限修改，否则只是文件夹权限修改

chown 用户名.用户组 文件路径

```
chown 用户名 文件路径 # 仅改变属主  
chown .用户名 文件路径 # 仅改变属组  
chgrp 用户组 文件路径 # 仅改变属组
```

使用的文件及文件目录最好使用绝对路径，以防止出错

基本权限ACL

Access control list 限制用户对文件访问

ACL是UGO的补充，或者可以说是加强版

查看文件ACL

命令 `getfacl` 获取文件acl

```
[ root@localhost tmp]# getfacl /home/test.txt
getfacl: Removing leading '/' from absolute path names
# file: home/test.txt
# owner: root
# group: root
user::rw-
user:alice:rw-
user:jack:---
group::r--
mask::rw-
other::r--
```

设置ACL

命令 `setfacl` 设置文件acl

- -m 设置对象权限
- -x 删除对象权限acl信息
- -d 设置为默认权限
- -b 擦除所有acl权限

`setfacl -m 对象: 对象名: 权限 文件路径`

`setfacl -x 对象: 对象名 文件路径`

设置facl之后，查看文件权限的时候，权限那边会出现一个+号，表达了是acl的权限补充

特殊权限

如果一个文件很重要，添加i权限让它不会被任何用户删除

特殊位

suid

当一个设置了SUID位的可执行文件被执行时，该文件将以所有者的身份运行，也就是说无论谁来执行这个文件，他都有文件所有者的特权。如果所有者是root的话，那么执行人就有超级用户的特权了。这时该位将变成一个安全漏洞，因此不要轻易设置该位。

sgid

当一个设置了SGID位的可执行文件运行时，该文件将具有所属组的特权，任意存取整个组所能使用的系统资源。若一个目录设置了SGID，则所有被复制到这个目录下的文件，其所属的组都会被重设为和这个目录一样，除非在复制文件时加上-p（preserve，保留文件属性）的参数，才能保留原来所属的群组设置。一般来说，SGID多用在特定的多人团队的项目开发上，在系统中用得较少。

sticky-bit

对一个文件设置了sticky-bit之后，尽管其他用户有写权限，也必须由属主执行删除、移动等操作。sticky-bit位要求操作系统既是在可执行程序退出后，仍要在内存中保留该程序的映象。这样做是为了节省大型程序的启动时间,但是会占用系统资源。因此设置该位，不如把程序写好。

SUID	SGID	sticky	二进制	八进制	说明
-	-	-	000	0	不设置特殊权限
-	-	t	001	1	只设置stiky
-	s	0	010	2	只设置sgid
s	-	-	100	4	只设置suid
-	s	t	011	3	设置sgid和stiky
s	-	t	101	5	设置suid和stiiky
s	s	-	110	6	设置suid和sgid
s	s	t	111	7	三者都设置

文件属性

```
chattr [ -RVf ] [ -v version ] [ mode ] files...
```

chattr 改变一个Linux文件系统上的文件属性

通用格式是：+--[aAcCdDeijsStTu]

a	只能附加数据
A	不修改访问时间
c	自动压缩文件
C	不执行写入时复制 (COW)
d	不能使用dump命令备份文件
D	更改同步保存
e	extent格式 (一种文件系统格式)
i	不能修改。不能删除或重命名，不能创建到该文件的链接，也不能向该文件写入数据。 只有超级用户或拥有 CAP_LINUX_IMMUTABLE 能力的进程才能设置或清除此属性
j	数据日志
s	安全删除
S	同步更新
t	
T	
u	文件被删除时，其内容会被保存，后面可以请求恢复

下面的只读属性，可以使用 lsattr列出，但不能被 chattr 修改：

- E: 压缩错误
- h: 巨大的文件
- I: 索引目录
- N: 内联数据
- X: 压缩原始访问
- Z: 压缩文件是脏的

并不是所有文件系统都支持所有标志；参考文件系统手册了解如btrfs(5), ext4(5), 和 xfs(5)文件格式的更多详情

umask

默认为0022

目录的默认权限是755

0777
-0022

755 系统为什么创建目录/文件夹的时候，权限时755.

文件的默认权限是644

系统为了保护自己，所以在创建的文件上。去掉了所有的执行。

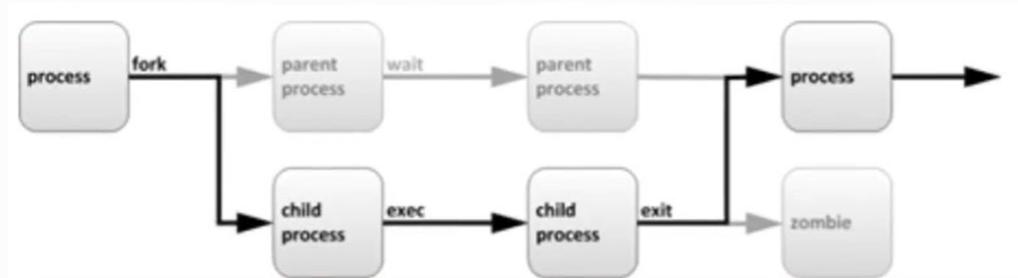
0755
- 0111

0644

Linux进程管理

虚拟文件系统--/proc/

进程的生命周期



父进程复制自己的地址空间 (fork) 创建一个新的进程

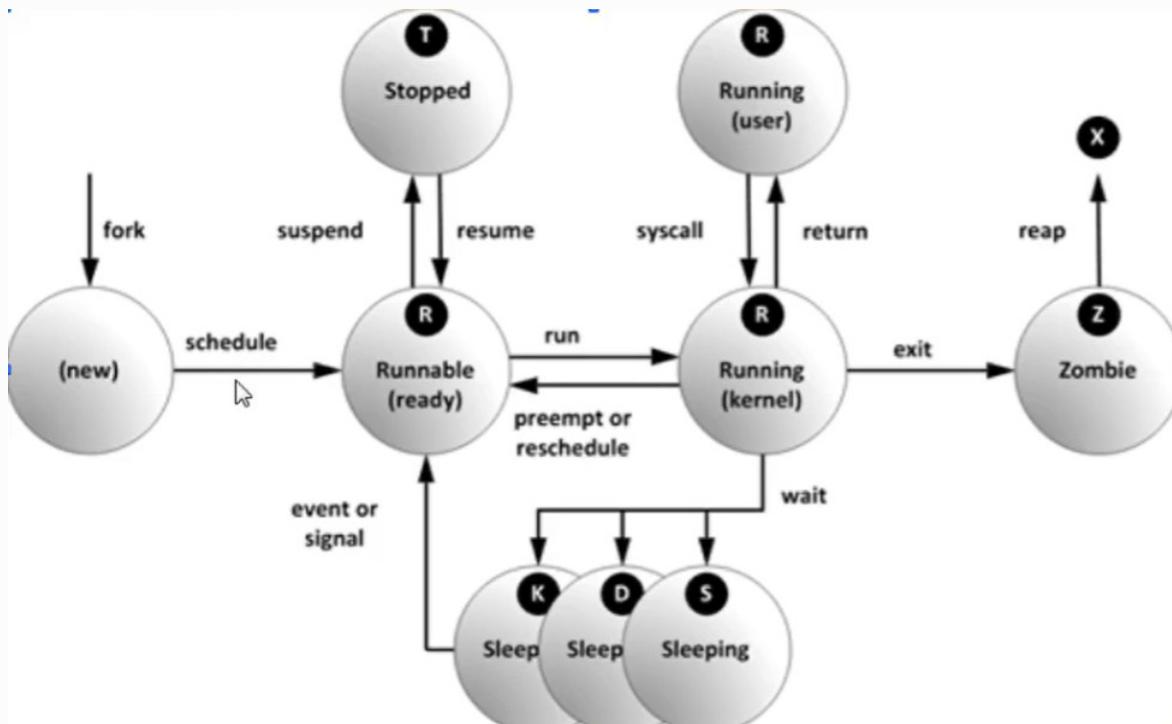
每个新进程分配一个唯一的进程ID（PID），满足后续查找跟踪，任何进程都可以创建子进程，所有进程都是第一个系统进程的后代

Centos5/6系统进程为init

Centos7系统进程为systemd

进程相关信息ps

USER	用户
PID	进程号
%CPU	CPU占用率
%MEM	内存占用率
VSZ	虚拟内存占用
RSS	使用的物理内存情况
TTY	使用的终端(多个终端同时访问时可以见到效果)
STAT	进程状态
START	启动时间
TIME	占用CPU时间
COMMAND	进程执行的命令行



静态查看进程ps

命令ps

- -a 显示当前终端的所有进程信息
- -u 以用户的格式显示进程的信息
- -x 显示后台进程运行的参数

常用

ps aux

ps aux --sort %cpu ps aux --sort -%cpu (-为降序)

ps axo 自定义的显示字段 自选择出现的字段 (自定义显示列)

动态查看进程top

命令top

- -d 控制刷新时间 top -d 3 (每隔三秒刷新一次, 默认为1)
- -p 查看某一个进程 top -p 进程号, 进程号 (查询某几个程序的情况)

top - 09:57:15 up 1:25, 2 users, load average: 0.00, 0.01, 0.05										
Tasks: 122 total, 1 running, 121 sleeping, 0 stopped, 0 zombie										
%Cpu(s): 0.0 us, 1.5 sy, 0.0 ni, 98.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
KiB Mem : 1863040 total, 1525640 free, 191692 used, 145708 buff/cache										
KiB Swap: 819196 total, 819196 free, 0 used. 1518868 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
9	root	20	0	0	0	0	S	6.2	0.0	0:01.42 rcu_sched
24	root	20	0	0	0	0	S	6.2	0.0	0:00.06 ksoftirqd/3
2010	root	20	0	158928	5608	4264	S	6.2	0.3	0:00.68 sshd
1	root	20	0	128412	6980	4144	S	0.0	0.4	0:02.81 systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01 kthreadd
4	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 kworker/0:0H
6	root	20	0	0	0	0	S	0.0	0.0	0:00.08 ksoftirqd/0
7	root	rt	0	0	0	0	S	0.0	0.0	0:00.21 migration/0
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00 rcu_bh
10	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 lru-add-drain
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.05 watchdog/0
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.07 watchdog/1
13	root	rt	0	0	0	0	S	0.0	0.0	0:00.21 migration/1
14	root	20	0	0	0	0	S	0.0	0.0	0:00.06 ksoftirqd/1
16	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 kworker/1:0H
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.07 watchdog/2
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.21 migration/2
19	root	20	0	0	0	0	S	0.0	0.0	0:00.30 ksoftirqd/2
21	root	0	-20	0	0	0	S	0.0	0.0	0:00.00 kworker/2:0H
22	root	rt	0	0	0	0	S	0.0	0.0	0:00.24 watchdog/3

上方是系统性能，下方是进程状态

top - 11:45:08 up 18:54, 4 users, load average: 0.05, 0.05, 0.05

程序名-系统时间 运行时间 登录用户数 CPU 负载 5分钟 10 15

Tasks: 176 total, 1 running, 175 sleeping, 0 stopped, 0 zombie

总进程数 运行数 1 睡眠数 175 停止数 0 僵死数 0

%Cpu(s): 0.0 us, 0.3 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

CPU 使用占比 us 用户 sy 系统 ni 优先级 id 空闲 wa 等待 hi 硬件 si 软件 st 虚拟机

在top指令生效的情况下，使用如下指令

M	按内存使用排序
P	按CPU使用排序
N	按PID大小排序
h ?	帮助
<	向前
>	向后
z	彩色 (使用数字调整)

信号控制进程Kill

```
[root@localhost ~]# kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGNALRM    15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO      30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

信号	编号	含义
SIGHUP	1	重新加载配置
SIGHNT	2	键盘终端ctrl+c
SIGQUIT	3	键盘退出ctrl+\
SIGKILL	9	强制终止, 无条件
SIGTERM	15	正常终止, 缺省信号,会保存
SIGCONT	18	继续
SIGSTOP	19	暂停
SIGTSTP	20	键盘暂停ctrl+Z

使用方式： `kill -信号编号 进程号`

进程优先级

查看、调整程序优先级，高优先级程序可以获得更多系统资源

优先范围和特性

nice优先级NI 及其范围为-20~19共40个级别

但是nice优先级并不是系统优先级

系统优先级为 PR，在实际的进程优先级会将NI优先级+20之后给纳入PR优先级

优先级级别越低，优先级越高

查看进程优先级

```
ps axo pid,command,nice --sort=-nice | head -5
```

更改NI优先级

默认情况下，启动进程时，会继承父进程的优先级

启动时调整

命令: nice -n (+/-) (数字) 运行的程序命令

已有调整

命令：先查询PID

```
renice (+/-)(数字) 程序PID
```

作业控制fg&bg

foreground(fg)

前台程序，在终端中运行的程序

把后台程序放入前台

命令 fg 程序序号

background(bg)

没有控制终端，不需要终端的交互，用户看不到，但是仍在运行

将程序放入后台

```
sleep 3 & 将sleep 3 这个程序放入后台，&符表示将程序放入后台运行
```

查看在后台的进程

命令: `jobs`

删除在后台的进程

`kill 进程号`

`kill %进程jobs内的序号`，注意其内的%

Linux管道符和重定向

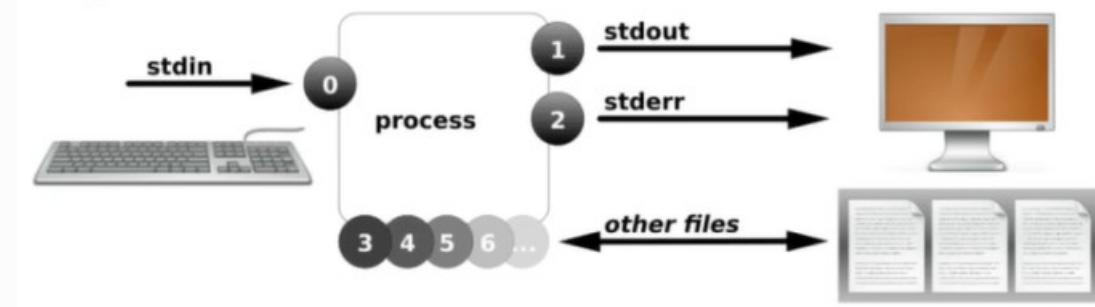
`< ><< > >> |`

重定向" <、>"

FD (0~255)

文件句柄，文件描述符，file description，进程使用文件时使用FD描述和管理相关文件

其实就是将路径和数字做关联，在使用时，无需时常考虑路径。类似于数字超链接



输出重定向

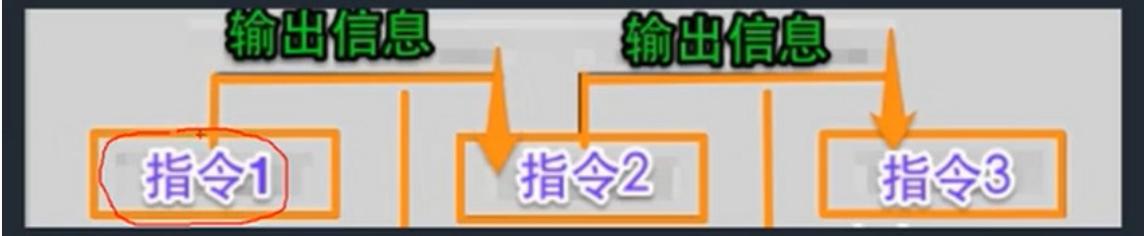
输入重定向

管道

piping

.管道命令可以将多条命令组合起来，一次性完成复杂的处理任务。

语法 command1 | command2 | command3 |...



tee(扩展)

使piping管道变成T字形管道，分流



命令: `cat /etc/passwd | tee file | tail -1`

参数传递

部分命令对 | 符不敏感，例如cp, rm等

使用xargs进行传递

例如: `cat delfile.txt | xargs rm -rfv`

Linux磁盘管理

命名方式

SATA--/dev/sda, /dev/sdb

/dev	设备文件目录
sda	文件
s	sata
d	disk
a	第一块硬盘 (后续跟着字母表顺序扩展)

新硬盘

分区 (MBR/GPT) ---->格式化/文件系统Filesystem----->挂载mount

查看磁盘信息

命令: `lsblk`

```
[root@localhost ~]# lsblk
NAME   MAJ:MIN RM  SIZE RTYPE MOUNTPOINT
sda      8:0    0  20G  disk
└─sda1   8:1    0 200M  part /boot
└─sda2   8:2    0 800M  part [SWAP]
└─sda3   8:3    0  19G  part /
sr0     11:0   1  4.5G rom
```

命令: `ls /dev/sd*`

```
[root@localhost ~]# ll /dev/sd*
brw-rw----. 1 root disk 8, 0 Dec  1 08:26 /dev/sda
brw-rw----. 1 root disk 8, 1 Dec  1 08:26 /dev/sda1
brw-rw----. 1 root disk 8, 2 Dec  1 08:26 /dev/sda2
brw-rw----. 1 root disk 8, 3 Dec  1 08:26 /dev/sda3
```

命令: `fdisk -l 目标磁盘文件`

```
[root@localhost ~]# fdisk -l /dev/sda

Disk /dev/sda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000d1c61

      Device Boot   Start     End   Blocks Id System
/dev/sda1  *       2048  411647  204800  83 Linux
/dev/sda2        411648 2050047  819200  82 Linux swap / Solaris
/dev/sda3        2050048 41943039 19946496  83 Linux
```

命令: `df`

- `-h` 人性化显示单位
- `-T` 显示文件系统类型

```
[root@localhost ~]# df -hT
Filesystem      Type      Size  Used  Avail Use% Mounted on
devtmpfs        devtmpfs  900M   0    900M  0% /dev
tmpfs          tmpfs     910M   0    910M  0% /dev/shm
tmpfs          tmpfs     910M  9.5M  901M  2% /run
tmpfs          tmpfs     910M   0    910M  0% /sys/fs/cgroup
/dev/sda3        xfs       20G  1.9G  18G  10% /
/dev/sda1        xfs      197M 121M  77M  62% /boot
tmpfs          tmpfs    182M   0   182M  0% /run/user/0
```

普通文件分区

创建分区

启动分区工具

命令：`fdisk 目标硬盘 (非系统盘)` 例：`fdisk /dev/sdb`

fdisk内使用命令

刷新分区表

命令：`partprobe 目标磁盘文件`

创建文件系统（格式化）

命令：`mkfs.ext4 目标磁盘文件分区` 例：`mkfs.ext4 /dev/sdb1`

make file system extend

挂载

创建挂载点（文件夹），一个分区一个挂载点，挂载点位置无所谓

命令：`mount -t 磁盘文件类型(ext4) 挂载目标磁盘分区文件 目标文件夹`

卸载

命令：`umount 分区挂载文件夹`

交换分区管理Swap

类似windows的虚拟内存，防止内存溢出OOM（Out of memory）

推荐 设置交换分区大小为内存的2倍
大于 4GB 而小于 16GB 内存的系统，最小需要 4GB 交换空间；
生产
大于 16GB 而小于 64GB 内存的系统，最小需要 8GB 交换空间；
大于 64GB 而小于 256GB 内存的系统，最小需要 16GB 交换空间。

分区准备

```
准备将 /dev/sde 磁盘，划分为 1G 分区为例  
划分分区后，将类型设置为 82（按 t！！！老铁）  
[root@server0 ~]# fdisk /dev/sde  
[root@server0 ~]# partprobe /dev/sde  
[root@server0 ~]# ll /dev/sde*  
brw-rw----. 1 root disk 253, 16 12月 6 10:18 /dev/sde  
brw-rw----. 1 root disk 253, 17 12月 6 10:18 /dev/sde1
```

其中第二部的类型设置现在由于系统改进，已经有默认值了，所以不需要改，但是保险起见还是改一下。

格式化

命令：`mkswap 目标磁盘分区文件`

挂载

命令：`swapon 目标磁盘分区文件`

卸载

命令：`swapoff 目标磁盘分区文件`

查看

命令：`free`

逻辑卷LVM

可以随意扩张大小，性能与普通磁盘无异，**只要卷组有空间，逻辑卷大小可以一直扩大**

`PV(Physic volume)` 物理卷

`VG(Volume Group)` 卷组

`LV(Logical Volume)` 逻辑卷

准备物理磁盘

创建PV

命令：`pvcreate 目标磁盘文件`

创建VG

命令：`vgcreate 卷组名 目标磁盘文件`

创建LV

命令：`lvcreate -L 逻辑卷大小 -n 逻辑卷名字 卷组名`

- `-L` 逻辑卷大小
- `-n` 逻辑卷名字
- 卷组名代表从哪个卷组中分配存储空间

格式化

命令：`mkfs -t ext4 /dev/卷组名/逻辑卷名`

挂载

创建文件夹之后，将文件夹挂载

命令：`mount /dev/卷组名/逻辑卷名 目标文件夹`

卸载

`umount...`

卷组扩容

命令：

1. `pvcreate 新添加的目标磁盘文件`

2. `vgextend 扩充的卷组 添加入卷组的目标磁盘文件`

逻辑卷扩容

LV扩容命令：`lvextend -L +2G /dev/卷组/逻辑卷`

扩容完逻辑卷之后需要对挂载的文件夹刷新一下

FS扩容命令：`resize2fs /dev/卷组/逻辑卷`

查看相关信息

`pvs` 查看物理卷信息

`vgs` 查看卷组信息

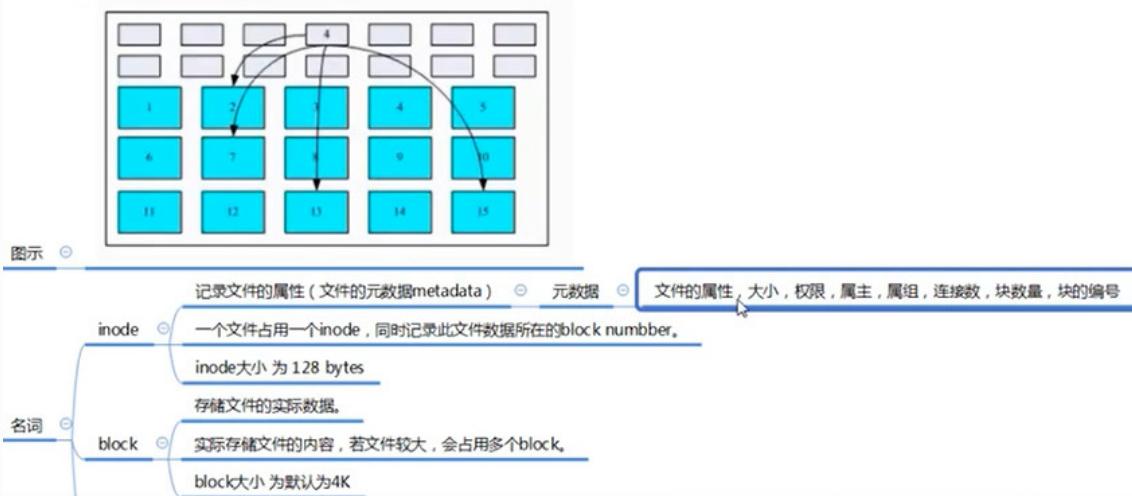
`lvs` 查看逻辑卷信息

文件系统

windows: FAT16 FAT32 NTF

Linux: Ext3 Ext4 XFS

Ext4:



RAID阵列

创建RAID

-C 创建RAID
/dev/md0 第一个RAID设备
-l5 RAID5
-n RAID成员的数量
-x 热备磁盘的数量
可用空间2G

```
[root@qianfeng ~]# mdadm -C /dev/md0 -l5 -n3 -x1 /dev/sd{d,e,f,g}
```

格式化、挂载

```
[root@qianfeng ~]# mkfs.xfs /dev/md0
[root@qianfeng ~]# mkdir /mnt/raid5
[root@qianfeng ~]# mount /dev/md0 /mnt/raid5
[root@qianfeng ~]# cp -rf /etc /mnt/raid5/etc1
```

查看RAID信息

```
[root@qianfeng ~]# mdadm -D /dev/md0 // -D 查看详细信息
```

坏盘移除

5. 模拟一块硬盘损坏，并移除

终端一：

```
[root@qianfeng ~]# watch -n 0.5 'mdadm -D /dev/md0 | tail' //watch持续查看
```

终端二：

```
[root@qianfeng ~]# mdadm /dev/md0 -f /dev/sde -r /dev/sde
//模拟坏了并移除
-f --fail
-r --remove
```

Linux文件查找及压缩

命令查找

命令：`which 命令查找` 例：`which ls`

文件查找

命令：

1. `find` 针对文件名

`find [path...] [options] [expression] [action]`

在某路径下找什么，怎么找，描述是啥

- `-name` 区分大小写的文件名
- `-iname` 不区分大小写的文件名
- `-size 大小`
 - `-size 5M` 文件等与5M的文件
 - `-size +5M` 文件大于5M的文件
 - `-size -5M` 文件小于5M的文件
- `-maxdepth 目录深度`
- `-user 属主名` (按文件属主查找)
- `-group 属组名` (按文件属组查找)
- `-type 文件类型` (按文件类型查找)
 - `l, b, d, -`
- `-perm 权限数字 -ls`

`find 查找目录 -name 文件名 -ok cp -rfv {} 目标目录 \;`

- `-ok` 代表后续接命令
- `{}` 代表查找到的文件

`find 查找目录 -name 文件名 -delete`

- 找到之后删除文件

2. locate 针对数据库例：

文件压缩解压

打包

命令：`tar -f 打包的包名 打包的文件、文件夹`

- -c create
- -f file

压缩

```
====打包，压缩=====
# tar -cf etc.tar /etc
# tar -czf etc-gzip.tar.gz /etc/          //z是gzip
# tar -cjf etc-bzip.tar.bz /etc/           //j是bzip
# tar -cJf etc-xzip.tar.xz /etc/          //J是xzip
观察三个包的体积。
# ll -h etc*
-rw-r--r--. 1 root root 11M 10月 14 10:07 etc-gzip.tar.gz
-rw-r--r--. 1 root root 8.9M 10月 14 10:08 etc-bzip.tar.bz
-rw-r--r--. 1 root root 7.6M 10月 14 10:08 etc-xzip.tar.xz
压缩速度和压缩体积成反比。
```

解压缩

命令：`tar -xf 包文件`

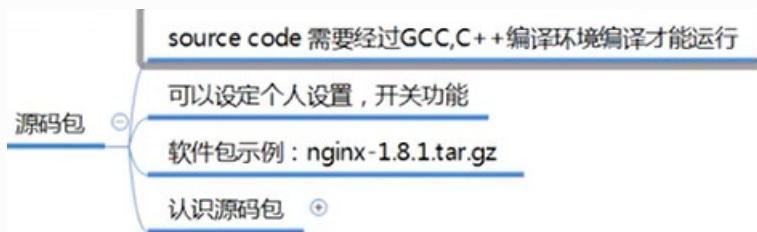
- -C解压重定向（解压文件放到另外一个文件夹中）例：`tar -xf 包文件 目标绝对路径`

Linux软件管理

RPM包

RPM Package Manager (原Red Hat Package Manager, 现在是一个递归缩写)
由 Red Hat 公司提出，被众多 Linux 发行版所采用
也称二进制 (binary code) 无需编译, 可以直接使用
无法设定个人设置，开关功能
软件包示例 (注意后缀) : mysql-community-common-5.7.12-1.el7.x86_64.rpm
认识RPM包

源码包



YUM

Yum (全称为 Yellow dog Updater, Modified)
是一个在Fedora和RedHat以及CentOS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件包，无须繁琐地一次次下载、安装。

本地yum源配置

· 观察repo文件

```
[root@localhost ~]# vim /etc/yum.repos.d/dvd.repo  
[dvd]  
name=dvd  
baseurl=file:///mnt/cdrom  
gpgcheck=0
```

/etc/yum.repos.d/是YUM下载地址库目录
[dvd]某一个库的名称，中括号[]是必须的
name=dvd是库的说明，name是必须的
baseurl=file:///mnt/cdrom下载库的具体地址
gpgcheck=0是关闭校验

· 备份--移除官方yum库

```
mv /etc/yum.repo.d/* /tmp
```

· 编写本地YUM库配置文件

```
vim /etc/yum.repo.d/dvd.repo  
[dvd]  
name=dvd  
baseurl=file:///mnt/cdrom  
gpgcheck=0
```

· 挂载安装光盘

```
[root@localhost ~]# mkdir /mnt/cdrom  
[root@localhost ~]# vim /root/.bashrc  
mount /dev/cdrom /mnt/cdrom  
重启后，使用ls /mnt/cdrom查看到光盘信息即可
```

写到文件中，开机加载

使用YUM管理程序

查看

命令： `yum list`

安装

命令： `yum install -y 软件报名`

· -y 自动yes

命令： `yum reinstall -y 软件报名` 重新安装

升级

命令： `yum -y update 包名`

卸载

命令： `yum -y remove 包名`

RPM

rpm安装，无法自动安装依赖，可能导致无法使用

查看

命令： `rpm -q 包名`

安装

命令： `rpm -ivh 包名`

使用包名的全称，包括后缀

卸载

命令: `rpm -evh 包名`

卸载软件, 使用软件名, **必须有后缀**

源码安装

. 准备编译工具

```
[root@localhost ~]# yum -y install gcc make zlib-devel pcre pcre-devel  
openssl-devel
```

. 解压及进入源码路径

`tar xvf 压缩文件名`

. 配置(`./configure`)

指定属组、属主、安装路径 (`--prefix=绝对路径`)

. 编译

`make`

. 安装

`make install`

Linux任务计划

在未来某个时间点, 计划一个任务运行

查看计划任务

命令: `atq`

一次性调度执行 `at`

schedule one-time tasks with at

命令: `at <timespace>`

```
at now +5min          # 从现在开始数5分钟  
at teatime tomorrow (teatime is 16:00)  
at noon +4days        #  
at 4:00 2022-01-01    # 具体时间点开始
```

例:

```
at now +1min  
at > useradd lsm  
at > (任务命令结束时, ctrl + D, 退出)
```

循环调度任务 cron

schedule resuring jobs with cron

cron的概念和crontab是不可分割的。
crontab是一个命令，常见于Unix和Linux的操作系统之中
用于设置周期性被执行的指令。
该命令从标准输入设备读取指令，并将其存放于“crontab”文件中，以供之后读取和执行。

内部会开放一个crond进程，进行命令的循环

查看此进程： `systemctl status crond.service`
`ps aux |grep crond`

计划任务的存储文件夹是 `/var/spool/cron`

命令： `crontab`

- `-e` 为当前用户编辑计划任务
- `-l` 列出当前用户的计划任务
- `-u username` 管理员可以管理其他用户的计划任务
- `-r` 删除当前用户的计划任务

```
# crontab -e
* * * * * command
分 时 日 月 周
(0~59) (0~23) (1~31) (1~12) (0~6, sunday=0 or 7)

5 1 15 3 * command # 每年3月15日1点5分执行command
5 1 15 * * command # 每年每月15号1点5分执行command
5 1 * * * command # 每年每月每日1点5分执行command
5 1 1,2,3 * * command # 每年每月的1, 2, 3号的1点5分执行command
5 1 5-9 * * command # 每年每月的5号~9号的1点5分执行command
0 * * * * command # 每个小时整点执行command
```

当月或日和周同时存在时，是U关系，即多个条件同时生效

Linux日志管理

系统日志rsyslog

系统日志管理，什么程序，产生了什么日志，放到了什么地方

系统操作有关的信息，如登录信息，程序启动和关闭信息，error信息

```
[root@localhost ~]# ps aux | grep rsyslog
root      1491  0.0  0.2 216424  5036 ?        Ssl  08:44   0:01 /usr/sbin/rsyslogd -n
```

d for daemon, 守护进程

应用程序日志

按着每个应用程序自己的方式产生、记录日志

常见的程序日志：

```
# tail -10 /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 /var/log/secure        //认证、安全
# tail /var/log/yum.log        //yum
# tail /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
# tail /var/log/mysqld.log      //MySQL
# tail /var/log/xferlog          //和访问FTP服务器相关
# tail /var/log/wtmp             //当前登录的用户(命令:w)
# tail /var/log/btmp             //最近登录的用户(命令last)
# tail /var/log/lastlog          //所有用户的登录情况(命令lastlog)
```

rsyslog配置

命令：`rpm -qc rsyslog` 列出与rsyslog相关的配置文件

- c是config的意思

`/etc/rsyslog.conf` rsyslogd的主配置文件(关键)
`/etc/sysconfig/rsyslog` rsyslogd相关文件，定义级别(了解一下)
`/etc/logrotate.d/syslog` 和日志轮转(切割)相关(任务二)

告诉rsyslog进程什么日志，应该存到哪里。

`# vim /etc/rsyslog.conf`

RULES : 即规则，是一套生成日志，以及存储日志的策略。
规则由设备+级别+存放位置组成。
RULES由FACILITY+LEVEL+FILE组成。
authpriv.* /var/log/secure (SSH信息)
mail.* -/var/log/maillog (发邮件)
cron.* /var/log/cron (创建任务)

```
# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

# File syncing capability is disabled by default. This feature is usually not required,
# not useful and an extreme performance hit
#$ActionFileEnableSync on
```

```

# Include all config files in /etc/rsyslog.d/
$IncludeConfig /etc/rsyslog.d/*.conf

# Turn off message reception via local log socket;
# local messages are retrieved through imjournal now.
$OmitLocalLogging on

# File to store the position in the journal
$IMJournalStateFile imjournal.state

##### RULES #####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                     /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none      /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                    /var/log/secure

# Log all the mail messages in one place.
mail.*                                         -/var/log/maillog
maillog

# Log cron stuff
cron.*                                         /var/log/cron

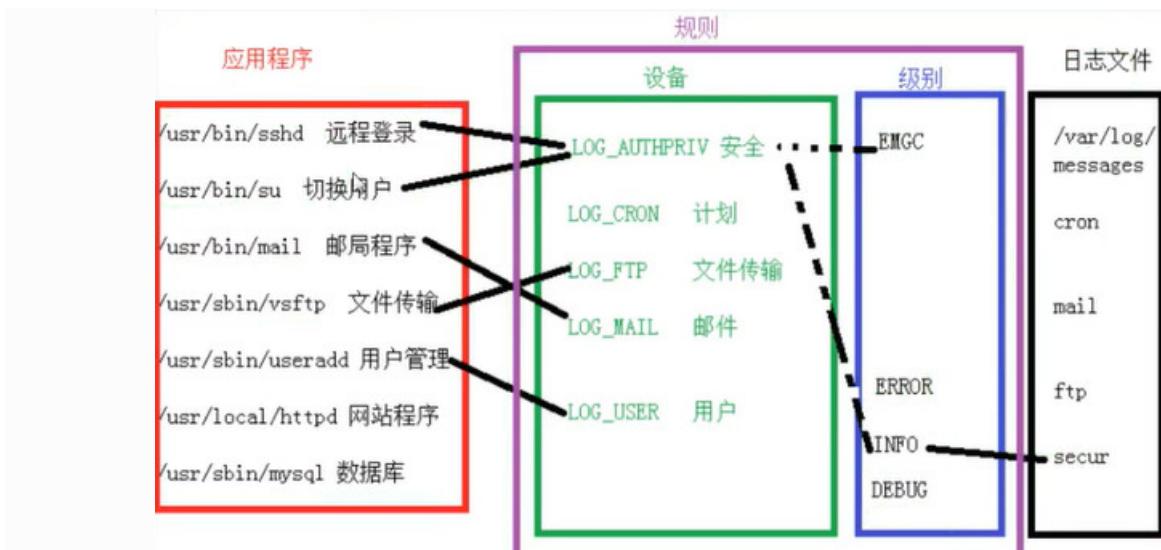
# Everybody gets emergency messages
*.emerg                                         :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                  /var/log/spooler

# Save boot messages also to boot.log
local7.*                                        /var/log/boot.log

```

其中的；代表并列



facility类型

facility在Linux中是对一类程序的说明



日志级别

从高到低排序

*代表所有级别

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，如磁盘空使用95%
LOG_CRIT	致命行为
LOG_ERR	错误行为
LOG_WARNING	警告信息
LOG_NOTICE	普通，重要的标准信息
LOG_INFO	标准信息
LOG_DEBUG	调试信息，排错所需，一般不建议使用 (*)

存放地址

文件路径前有一个 `-`，代表异步进行

可修改，修改完需要重启服务或者重启计算机

日志轮转logrotate

对日志信息进行处理，以节省磁盘空间，抛弃无效数据

主文件 : /etc/logrotate.conf (决定每个日志文件如何轮转)

子文件夹 : /etc/logrotate.d/*

logrotate配置规则

主配置文件

```
# see "man logrotate" for details
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# create new (empty) log files after rotating old ones
create
# use date as a suffix of the rotated file
dateext
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own wtmp and btmp -- we'll rotate them here
/var/log/wtmp {
    monthly                      # 轮转周期
    create 0664 root utmp      # 轮转后创建新文件、并设置权限为0664 属主 属组
    minsize 1M                   # 最小达到1M才进行轮转
    rotate 1                     # 保留一份
}
/var/log/btmp {
    missingok                    # 丢失不提示 missing is ok 文件不重要
    monthly
    create 0600 root utmp
    rotate 1
}
```

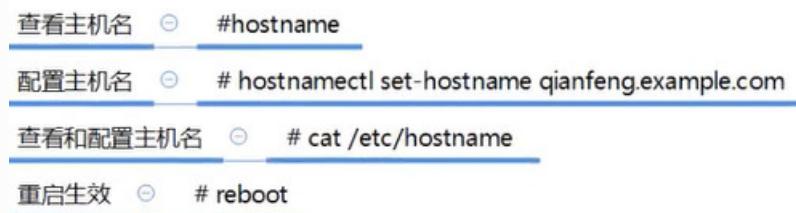
weekly	轮转周期，一周一轮转
rotate4	轮转保留4份
create 0600 root utmp	轮转后创建新文件 文件权限 文件属主 文件数组
dateext	使用日期作为后缀
compress	是否压缩
include 文件夹	包含该目录下的配置文件（还有一些其他配置在别的地方，这个地方在哪呢？）
notifempty	空文件不进行轮转
minsize	最小进行轮转的文件大小
missingok	丢失不提示，文件不重要
maxsize	文件最大达到多大，达到最大即刻轮转

轮转配置分全局配置和单文件配置，全局配置是默认，单文件是实际，如果单文件没有要求，则看全局文件。

轮转配置主要有两大维度，时间及大小，是否是达到时间进行切割，还是达到大小进行切割。如果两个唯独同时设置，则需要同时满足两个条件，为“与”关系

Linux网络管理

更改主机名



Linux文件服务

ftp Server

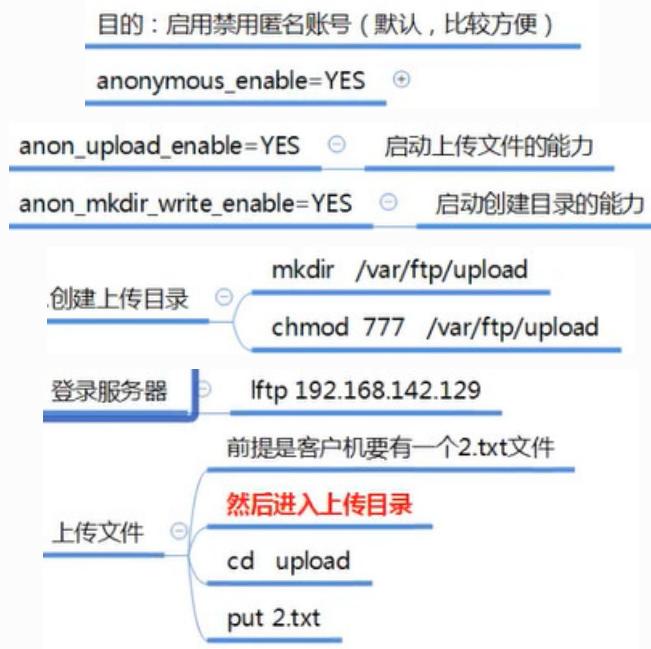
File Transfer protocol 是TCP/IP协议组中的协议之一

提供文件共享服务

软件包为vsftpd

/var/ftp	分发的文件路径
ftp://服务器IP地址	访问路径

文件上传



NFS

Network File System，网络文件系统，NFS的客户端主要为Linux，支持多节点同时挂载并发写入

将一个服务器作为存储专用服务器，集中存储文件

其它网站服务器等从存储服务器内部查询文件

安装NFS服务器--存储端

安装nfs服务

命令: `yum -y install nfs-utils`

在本地创建一个文件夹作为共享文件夹

配置nfs服务

在 `文件/etc/exports` 文件中写入: `本地共享文件夹 共享网段`

启动服务，开机自启动

命令: `systemctl start nfs-server` `systemctl enable nfs-server`

例:

```
# yum -y install nfs-utils  
# mkdir /webdata ② 存储网站代码  
# echo "nfs test..." > /webdata/index.html ③ 放置测试页面  
[root@nas ~]# vim /etc/exports  
/webdata 192.168.122.0/24(rw)
```

/webdata 是发布资源的目录

```
[root@nas ~]# systemctl start nfs-server  
[root@nas ~]# systemctl enable nfs-server
```

export -v 检查输出的文件夹

```
[root@nas ~]# exportfs -v ④ -v 检查输出的目录  
/webdata  
192.168.122.0/24(rw,wdelay,no_root_squash,no_subtree_check,sec=sys,rw,secure,no_r
```

安装NFS服务--WEB服务器

安装nfs服务

命令：`yum -y install nfs-utils httpd`

httpd是用来测试的

启动及开机启动

显示挂载情况

命令：`showmount -e 目标nfs存储服务器`

挂载

命令：`mount -t nfs 目标nfs存储服务器:存储服务器内的共享文件夹 本地挂载文件夹`

查看挂载

命令：`df`

例：

```
[root@web1 ~]# yum -y install nfs-utils httpd  
[root@web1 ~]# systemctl start httpd  
[root@web1 ~]# systemctl enable httpd  
[root@web1 ~]# showmount -e 192.168.142.133 ⑤ // 查询NFS服务器可用目录  
Export list for 192.168.142.133  
/webdata 192.168.142.0/24
```

```
[root@web1 ~]# mount -t nfs 192.168.142.133:/webdata /var/www/html/  
[root@web1 ~]# df  
192.168.122.152:/webdata 7923136 692416 6821568 10% /var/www/html  
[root@web1 ~]# ls /var/www/html/  
index.html
```

ssh

安装-->启动及自启动



Linux防火墙

firewall

保护互联网对服务器的影响

systemctl stop firewalld 停止

systemctl disable firewalld 禁止开机自启动

systemctl status firewalld 查看程序状态

selinux

保护服务器内部程序对内部文件的访问

centos7以后的程序只能访问其内部文件，对于其他文件都不准访问

状态查看

getenforce

- enforcing 开启
- permissive 放行
- disable 关闭

临时关闭

命令: `setenforce 0`

永久关闭 (修改配置文件)

文件: `/etc/selinux/config`

将SELINUX改为disable

SHELL编程

shell的分类

bash、ash、ksh、csh等

bash

bash-complete 自动补全程序 (好像已经自带了)

命令历史记录

历史命令是保存在 `.bash_history` 隐藏文件中的

命令: `history`

- 数字 显示最近的几条命令
- -d n 删除第n条历史

命令: `!某一条历史记录编号` 重新运行该编号的程序

- !-n 执行倒数第几条命令
- !! 执行最后一天命令

bash环境文件

文件	描述	图形界面	文本界面	su -l	su
/etc/profile	全局配置，所有用户登陆的时候都会读取	1	2	2	
/etc/bashrc(/etc/bash.bashrc)	全局配置文件，bash执行的时候读取	3	1	1	1
~/.profile		2			
~/.bash_login					
~/.bash_profile			3	3	
~/.bashrc		4			2
~/.bash_logout					

bash特性

命令大全

1、ls命令

就是 list 的缩写，通过 ls 命令不仅可以查看 linux 文件夹包含的文件，而且可以查看文件权限(包括目录、文件夹、文件权限)查看目录信息等等。

常用参数搭配：

```
ls /文件夹名 列出文件夹内非隐藏文件  
ls -a 列出目录所有文件，包含以.开始的隐藏文件  
ls -A 列出除.及..的其它文件  
ls -r 反序排列  
ls -t 以文件修改时间排序  
ls -S 以文件大小排序  
ls -h 以易读大小显示  
ls -l 除了文件名之外，还将文件的权限、所有者、文件大小等信息详细列出来
```

实例：

(1) 按易读方式按时间反序排序，并显示文件详细信息

```
ls -lhrt
```

(2) 按大小反序显示文件详细信息

```
ls -lrs
```

(3)列出当前目录中所有以"t"开头的目录的详细内容

```
ls -l t*
```

(4)列出文件绝对路径（不包含隐藏文件）

```
ls | sed "s:^:`pwd`:/"
```

(5)列出文件绝对路径（包含隐藏文件）

```
find $pwd -maxdepth 1 | xargs ls -ld
```

2、cd 命令

cd(changeDirectory) 命令语法：

```
cd [目录名]
```

说明：切换当前目录至 dirName。

实例：

(1) 进入要目录

```
cd /
```

(2) 进入 "home" 目录

```
cd ~
```

(3) 进入上一次工作路径

```
cd -
```

(4) 把上个命令的参数作为cd参数使用。

```
cd !$
```

3、pwd 命令

pwd 命令用于查看当前工作目录路径。

实例：

(1) 查看当前路径

```
pwd
```

(2) 查看软链接的实际路径

```
pwd -P
```

4、mkdir 命令

mkdir 命令用于创建文件夹。

可用选项：

- **-m**: 对新建目录设置存取权限，也可以用 chmod 命令设置；
- **-p**: 可以是一个路径名称。此时若路径中的某些目录尚不存在,加上此选项后，系统将自动建立好那些尚不在的目录，即一次可以建立多个目录。

实例：

(1) 当前工作目录下创建名为 t 的文件夹

```
mkdir t
```

(2) 在 tmp 目录下创建路径为 test/t1/t 的目录，若不存在，则创建：

```
mkdir -p /tmp/test/t1/t
```

5、rm 命令

删除一个目录中的一个或多个文件或目录，如果没有使用 **-r** 选项，则 rm 不会删除目录。如果使用 rm 来删除文件，通常仍可以将该文件恢复原状。

```
rm [选项] 文件...
```

实例：

(1) 删除任何 .log 文件，删除前逐一询问确认：

```
rm -i *.log
```

(2) 删除 test 子目录及子目录中所有档案删除，并且不用一一确认：

```
rm -rf test
```

(3) 删除以 -f 开头的文件

```
rm -- -f*
```

6、 rmdir 命令

从一个目录中删除一个或多个子目录项，删除某目录时也必须具有对其父目录的写权限。

注意：不能删除非空目录

实例：

(1) 当 parent 子目录被删除后使它也成为空目录的话，则顺便一并删除：

```
rmdir -p parent/child/child11
```

7、 mv 命令

移动文件或修改文件名，根据第二参数类型（如目录，则移动文件；如为文件则重命名该文件）。

当第二个参数为目录时，第一个参数可以是多个以空格分隔的文件或目录，然后移动第一个参数指定的多个文件到第二个参数指定的目录中。

实例：

(1) 将文件 test.log 重命名为 test1.txt

```
mv test.log test1.txt
```

(2) 将文件 log1.txt,log2.txt,log3.txt 移动到根的 test3 目录中

```
mv log1.txt log2.txt log3.txt /test3
```

(3) 将文件 file1 改名为 file2，如果 file2 已经存在，则询问是否覆盖

```
mv -i log1.txt log2.txt
```

(4) 移动当前文件夹下的所有文件到上一级目录

```
mv * ../
```

8、 cp 命令

将源文件复制至目标文件，或将多个源文件复制至目标目录。

注意：命令行复制，如果目标文件已经存在会提示是否覆盖，而在 shell 脚本中，如果不加 -i 参数，则不会提示，而是直接覆盖！

```
-i 提示  
-r 复制目录及目录内所有项目  
-a 复制的文件与原文件时间一样
```

实例：

- (1) 复制 a.txt 到 test 目录下，保持原文件时间，如果原文件存在提示是否覆盖。

```
cp -ai a.txt test
```

- (2) 为 a.txt 建议一个链接（快捷方式）

```
cp -s a.txt link_a.txt
```

9、cat 命令

cat 主要有三大功能：

- 1.一次显示整个文件：

```
cat filename
```

- 2.从键盘创建一个文件：

```
cat > filename
```

只能创建新文件，不能编辑已有文件。

- 3.将几个文件合并为一个文件：

```
cat file1 file2 > file
```

- -b 对非空输出行号
- -n 输出所有行号

实例：

- (1) 把 log2012.log 的文件内容加上行号后输入 log2013.log 这个文件里

```
cat -n log2012.log log2013.log
```

- (2) 把 log2012.log 和 log2013.log 的文件内容加上行号（空白行不加）之后将内容附加到 log.log 里

```
cat -b log2012.log log2013.log log.log
```

- (3) 使用 here doc 生成新文件

```
cat >log.txt <<EOF
>Hello
>World
>PWD=$(pwd)
>EOF
ls -l log.txt
cat log.txt
Hello
World
PWD=/opt/soft/test
```

(4) 反向列示

```
tac log.txt
PWD=/opt/soft/test
World
Hello
```

10、more 命令

功能类似于 cat, more 会以一页一页的显示方便使用者逐页阅读，而最基本的指令就是按空白键（space）就往下一页显示，按 b 键就会往回（back）一页显示。

命令参数：

+n	从第 n 行开始显示
-n	定义屏幕大小为n行
+/pattern	在每个档案显示前搜寻该字串 (pattern) , 然后从该字串前两行之后开始显示
-c	从顶部清屏, 然后显示
-d	提示“Press space to continue, 'q' to quit (按空格键继续, 按q键退出) ”, 禁用响铃功能
-l	忽略Ctrl+l (换页) 字符
-p	通过清除窗口而不是滚屏来对文件进行换页, 与-c选项相似
-s	把连续的多个空行显示为一行
-u	把文件内容中的下画线去掉

常用操作命令：

Enter	向下 n 行, 需要定义。默认为 1 行
Ctrl+F	向下滚动一屏
空格键	向下滚动一屏
Ctrl+B	返回上一屏
=	输出当前行的行号
:f	输出文件名和当前行的行号
V	调用vi编辑器
!命令	调用Shell, 并执行命令
q	退出more

实例：

(1) 显示文件中从第3行起的内容

```
more +3 text.txt
```

(2) 在所列出文件目录详细信息，借助管道使每次显示 5 行

```
ls -l | more -5
```

按空格显示下 5 行。

11、less 命令

less 与 more 类似，但使用 less 可以随意浏览文件，而 more 仅能向前移动，却不能向后移动，而且 less 在查看之前不会加载整个文件。

常用命令参数：

```
-i 忽略搜索时的大小写  
-N 显示每行的行号  
-o <文件名> 将less 输出的内容在指定文件中保存起来  
-s 显示连续空行为一行  
/字符串：向下搜索“字符串”的功能  
?字符串：向上搜索“字符串”的功能  
n: 重复前一个搜索（与 / 或 ? 有关）  
N: 反向重复前一个搜索（与 / 或 ? 有关）  
-x <数字> 将“tab”键显示为规定的数字空格  
b 向后翻一页  
d 向后翻半页  
h 显示帮助界面  
Q 退出less 命令  
u 向前滚动半页  
y 向前滚动一行  
空格键 滚动一行  
回车键 滚动一页  
[pagedown]: 向下翻动一页  
[pageup]: 向上翻动一页
```

实例：

(1) ps 查看进程信息并通过 less 分页显示

```
ps -aux | less -N
```

(2) 查看多个文件

```
less 1.log 2.log
```

可以使用 n 查看下一个，使用 p 查看前一个。

12、head 命令

head 用来显示档案的开头至标准输出中，默认 head 命令打印其相应文件的开头 10 行。

常用参数：

```
-n<行数> 显示的行数（行数为复数表示从最后向前数）
```

实例：

(1) 显示 1.log 文件中前 20 行

```
head 1.log -n 20
```

(2) 显示 1.log 文件前 20 字节

```
head -c 20 log2014.log
```

(3) 显示 t.log 最后 10 行

```
head -n -10 t.log
```

13、tail 命令

用于显示指定文件末尾内容，不指定文件时，作为输入信息进行处理。常用查看日志文件。

常用参数：

-f 循环读取（常用于查看递增的日志文件）

-n<行数> 显示行数（从后向前）

(1) 循环读取逐渐增加的文件内容

```
ping 127.0.0.1 > ping.log &
```

后台运行：可使用 jobs -l 查看，也可使用 fg 将其移到前台运行。

```
tail -f ping.log
```

(查看日志)

14、which 命令

在 linux 要查找某个文件，但不知道放在哪里了，可以使用下面的一些命令来搜索：

which 查看可执行文件的位置。

whereis 查看文件的位置。

locate 配合数据库查看文件位置。

find 实际搜寻硬盘查询文件名称。

which 是在 PATH 就是指定的路径中，搜索某个系统命令的位置，并返回第一个搜索结果。使用 which 命令，就可以看到某个系统命令是否存在，以及执行的到底是哪一个位置的命令。

常用参数：

-n 指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。

实例：

(1) 查看 ls 命令是否存在，执行哪个

```
which ls
```

(2) 查看 which

```
which which
```

(3) 查看 cd

```
which cd (显示不存在，因为 cd 是内建命令，而 which 查找显示是 PATH 中的命令)
```

查看当前 PATH 配置：

```
echo $PATH
```

或使用 env 查看所有环境变量及对应值

15、whereis 命令

whereis 命令只能用于程序名的搜索，而且只搜索二进制文件（参数-b）、man说明文件（参数-m）和源代码文件（参数-s）。如果省略参数，则返回所有信息。whereis 及 locate 都是基于系统内建的数据库进行搜索，因此效率很高，而find则是遍历硬盘查找文件。

常用参数：

- b 定位可执行文件。
- m 定位帮助文件。
- s 定位源代码文件。
- u 搜索默认路径下除可执行文件、源代码文件、帮助文件以外的其它文件。

实例：

(1) 查找 locate 程序相关文件

```
whereis locate
```

(2) 查找 locate 的源码文件

```
whereis -s locate
```

(3) 查找 locate 的帮助文件

```
whereis -m locate
```

16、locate 命令

locate 通过搜寻系统内建文档数据库达到快速找到档案，数据库由 updatedb 程序来更新，updatedb 是由 cron daemon 周期性调用的。默认情况下 locate 命令在搜寻数据库时比由整个由硬盘资料来搜寻资料来得快，但较差劲的是 locate 所找到的档案若是最近才建立或刚更名的，可能会找不到，在内定值中，updatedb 每天会跑一次，可以由修改 crontab 来更新设定值 (etc/crontab)。

locate 与 find 命令相似，可以使用如 *、? 等进行正则匹配查找

常用参数：

- l num (要显示的行数)
- f 将特定的档案系统排除在外，如将proc排除在外
- r 使用正则运算式做为寻找条件

实例：

(1) 查找和 pwd 相关的所有文件(文件名中包含 pwd)

```
locate pwd
```

(2) 搜索 etc 目录下所有以 sh 开头的文件

```
locate /etc/sh
```

(3) 查找 /var 目录下，以 reason 结尾的文件

```
locate -r '^/var.*reason$' (其中.表示一个字符，*表示任务多个；.*表示任意多个字符)
```

17、find 命令

用于在文件树中查找文件，并作出相应的处理。

命令格式：

```
find pathname -options [-print -exec -ok ...]
```

命令参数：

- pathname: find 命令所查找的目录路径。例如用.来表示当前目录，用/来表示系统根目录。
- print: find 命令将匹配的文件输出到标准输出。
- exec: find 命令对匹配的文件执行该参数所给出的 shell 命令。
相应命令的形式为 'command' { } \;，注意{ }和\;之间的空格。
- ok: 和-exec 的作用相同，只不过以一种更为安全的模式来执行该参数所给出的 shell 命令
在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

命令选项：

- name 按文件名查找文件
- perm 按文件权限查找文件
- user 按文件属主查找文件
- group 按照文件所属的组来查找文件。

```
-type   查找某一类型的文件，诸如：  
      b - 块设备文件  
      d - 目录  
      c - 字符设备文件  
      l - 符号链接文件  
      p - 管道文件  
      f - 普通文件  
  
-size n :[c]  查找文件长度为n块文件，带有c时表文件字节大小  
-amin n    查找系统中最后N分钟访问的文件  
-atime n   查找系统中最后n*24小时访问的文件  
-cmin n   查找系统中最后N分钟被改变文件状态的文件  
-ctime n   查找系统中最后n*24小时被改变文件状态的文件  
-mmin n   查找系统中最后N分钟被改变文件数据的文件  
-mtime n   查找系统中最后n*24小时被改变文件数据的文件  
(用减号-来限定更改时间在距今n日以内的文件，而用加号+来限定更改时间在距今n日以前的文件。 )  
-maxdepth n 最大查找目录深度  
-prune 选项来指出需要忽略的目录。在使用-prune选项时要当心，因为如果你同时使用了-depth选项，那么-prune选项  
就会被find命令忽略  
-newer 如果希望查找更改时间比某个文件新但比另一个文件旧的所有文件，可以使用-newer选项
```

实例：

(1) 查找 48 小时内修改过的文件

```
find -atime -2
```

(2) 在当前目录查找 以 .log 结尾的文件。. 代表当前目录

```
find ./ -name '*.log'
```

(3) 查找 /opt 目录下 权限为 777 的文件

```
find /opt -perm 777
```

(4) 查找大于 1K 的文件

```
find -size +1000c
```

查找等于 1000 字符的文件

```
find -size 1000c
```

-exec 参数后面跟的是 command 命令，它的终止是以 ; 为结束标志的，所以这句命令后面的分号是不可缺少的，
考虑到各个系统中分号会有不同的意义，所以前面加反斜杠。{} 花括号代表前面find查找出的文件名。

实例：

(5) 在当前目录中查找更改时间在10日以前的文件并删除它们(无提醒)

```
find . -type f -mtime +10 -exec rm -f {} \;
```

(6) 当前目录中查找所有文件名以.log结尾、更改时间在5日以上的文件，并删除它们，只不过在删除之前先给出提示。按y键删除文件，按n键不删除

```
find . -name '*.log' mtime +5 -ok -exec rm {} \;
```

(7) 当前目录下查找文件名以 passwd 开头，内容包含 "pkg" 字符的文件

```
find . -f -name 'passwd*' -exec grep "pkg" {} \;
```

(8) 用 exec 选项执行 cp 命令

```
find . -name '*.log' -exec cp {} test3 \;
```

-xargs find 命令把匹配到的文件传递给 xargs 命令，而 xargs 命令每次只获取一部分文件而不是全部，不像 -exec 选项那样。这样它可以先处理最先获取的一部分文件，然后是下一批，并如此继续下去。

实例：

(9) 查找当前目录下每个普通文件，然后使用 xargs 来判断文件类型

```
find . -type f -print | xargs file
```

(10) 查找当前目录下所有以 js 结尾的并且其中包含 'editor' 字符的普通文件

```
find . -type f -name "*.js" -exec grep -lF 'ueditor' {} \;
find -type f -name '*.js' | xargs grep -lF 'editor'
```

(11) 利用 xargs 执行 mv 命令

```
find . -name "*.log" | xargs -i mv {} test4
```

(12) 用 grep 命令在当前目录下的所有普通文件中搜索 hostnames 这个词，并标出所在行：

```
find . -name \*(转义) -type f -print | xargs grep -n 'hostnames'
```

(13) 查找当前目录中以一个小写字母开头，最后是 4 到 9 加上.log 结束的文件：

```
find . -name '[a-z]*[4-9].log' -print
```

(14) 在 test 目录查找不在 test4 子目录查找

```
find test -path 'test/test4' -prune -o -print
```

(15) 实例1：查找更改时间比文件 log2012.log 新但比文件 log2017.log 旧的文件

```
find -newer log2012.log ! -newer log2017.log
```

使用 depth 选项：

depth 选项可以使 find 命令向磁带上备份文件系统时，希望首先备份所有的文件，其次再备份子目录中的文件。

实例：find 命令从文件系统的根目录开始，查找一个名为 CON.FILE 的文件。它将首先匹配所有的文件然后再进入子目录中查找

```
find / -name "CON.FILE" -depth -print
```

18、chmod 命令

用于改变 linux 系统文件或目录的访问权限。用它控制文件或目录的访问权限。该命令有两种用法。一种是包含字母和操作符表达式的文字设定法；另一种是包含数字的数字设定法。

每一文件或目录的访问权限都有三组，每组用三位表示，分别为文件属主的读、写和执行权限；与属主同组的用户的读、写和执行权限；系统中其他用户的读、写和执行权限。可使用 ls -l test.txt 查找。

以文件 log2012.log 为例：

```
-rw-r--r-- 1 root root 296K 11-13 06:03 log2012.log
```

第一列共有 10 个位置，第一个字符指定了文件类型。在通常意义上，一个目录也是一个文件。如果第一个字符是横线，表示是一个非目录的文件。如果是 d，表示是一个目录。从第二个字符开始到第十个 9 个字符，3 个字符一组，分别表示了 3 组用户对文件或者目录的权限。权限字符用横线代表空许可，r 代表只读，w 代表写，x 代表可执行。

常用参数：

```
-c 当发生改变时，报告处理信息  
-R 处理指定目录以及其子目录下所有文件
```

权限范围：

```
u : 目录或者文件的当前的用户  
g : 目录或者文件的当前的群组  
o : 除了目录或者文件的当前用户或群组之外的用户或者群组  
a : 所有的用户及群组
```

权限代号：

```
r : 读权限，用数字4表示  
w : 写权限，用数字2表示  
x : 执行权限，用数字1表示  
- : 删权权限，用数字0表示  
s : 特殊权限
```

实例：

(1) 增加文件 t.log 所有用户可执行权限

```
chmod a+x t.log
```

(2) 撤销原来所有的权限，然后使拥有者具有可读权限，并输出处理信息

```
chmod u=r t.log -c
```

(3) 给 file 的属主分配读、写、执行(7)的权限，给file的所在组分配读、执行(5)的权限，给其他用户分配执行(1)的权限

```
chmod 751 t.log -c (或者: chmod u=rwx,g=rx,o=x t.log -c)
```

(4) 将 test 目录及其子目录所有文件添加可读权限

```
chmod u+r,g+r,o+r -R text/ -c
```

19、tar 命令

用来压缩和解压文件。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。

弄清两个概念：打包和压缩。打包是指将一大堆文件或目录变成一个总的文件；压缩则是将一个大的文件通过一些压缩算法变成一个小文件

常用参数：

- c 建立新的压缩文件
- f 指定压缩文件
- r 添加文件到已经压缩文件包中
- u 添加改了和现有的文件到压缩包中
- x 从压缩包中抽取文件
- t 显示压缩文件中的内容
- z 支持gzip压缩
- j 支持bzip2压缩
- Z 支持compress解压文件
- v 显示操作过程

有关 gzip 及 bzip2 压缩:

gzip 实例: 压缩 gzip fileName .tar.gz 和.tgz 解压: gunzip filename.gz 或 gzip -d filename.gz
 对应: tar zcvf filename.tar.gz tar zxvf filename.tar.gz

bz2实例: 压缩 bzip2 -z filename .tar.bz2 解压: bunzip filename.bz2或bzip -d filename.bz2
 对应: tar jcvf filename.tar.gz 解压: tar jxvf filename.tar.bz2

实例：

(1) 将文件全部打包成 tar 包

```
tar -cvf log.tar 1.log,2.log 或tar -cvf log.*
```

(2) 将 /etc 下的所有文件及目录打包到指定目录，并使用 gz 压缩

```
tar -zcvf /tmp/etc.tar.gz /etc
```

(3) 查看刚打包的文件内容 (一定加z，因为是使用 gzip 压缩的)

```
tar -ztvf /tmp/etc.tar.gz
```

(4) 要压缩打包 /home, /etc，但不要 /home/dmtsai

```
tar --exclude /home/dmtsai -zcvf myfile.tar.gz /home/* /etc
```

20、chown 命令

chown 将指定文件的拥有者改为指定的用户名或组，用户可以是用户名或者用户 ID；组可以是组名或者组 ID；文件是以空格分开的要改变权限的文件列表，支持通配符。

```
-c 显示更改的部分的信息  
-R 处理指定目录及子目录
```

实例：

(1) 改变拥有者和群组 并显示改变信息

```
chown -c mail:mail log2012.log
```

(2) 改变文件群组

```
chown -c :mail t.log
```

(3) 改变文件夹及子文件目录属主及属组为 mail

```
chown -cR mail: test/
```

21、df 命令

显示磁盘空间使用情况。获取硬盘被占用了多少空间，目前还剩下多少空间等信息，如果没有文件名被指定，则所有当前被挂载的文件系统的可用空间将被显示。默认情况下，磁盘空间将以 1KB 为单位进行显示，除非环境变量 `POSIXLY_CORRECT` 被指定，那样将以512字节为单位进行显示：

```
-a 全部文件系统列表  
-h 以方便阅读的方式显示信息  
-i 显示inode信息  
-k 区块为1024字节  
-l 只显示本地磁盘  
-T 列出文件系统类型
```

实例：

(1) 显示磁盘使用情况

```
df -l
```

(2) 以易读方式列出所有文件系统及其类型

```
df -haT
```

22、du 命令

du 命令也是查看使用空间的，但是与 df 命令不同的是 Linux du 命令是对文件和目录磁盘使用的空间的查看：

命令格式：

```
du [选项] [文件]
```

常用参数：

-a 显示目录中所有文件大小
-k 以KB为单位显示文件大小
-m 以MB为单位显示文件大小
-g 以GB为单位显示文件大小
-h 以易读方式显示文件大小
-s 仅显示总计
-c或--total 除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和

实例：

(1) 以易读方式显示文件夹内及子文件夹大小

```
du -h scf/
```

(2) 以易读方式显示文件夹内所有文件大小

```
du -ah scf/
```

(3) 显示几个文件或目录各自占用磁盘空间的大小，还统计它们的总和

```
du -hc test/ scf/
```

(4) 输出当前目录下各个子目录所使用的空间

```
du -hc --max-depth=1 scf/
```

23、ln 命令

功能是为文件在另外一个位置建立一个同步的链接，当在不同目录需要该问题时，就不需要为每一个目录创建同样的文件，通过 ln 创建的链接（link）减少磁盘占用量。

链接分类：软链接及硬链接

软链接：

- 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:

- 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

需要注意:

- 第一：ln命令会保持每一处链接文件的同步性，也就是说，不论你改动了哪一处，其它的文件都会发生相同的变化；
- 第二：ln的链接又分软链接和硬链接两种，软链接就是ln -s 源文件 目标文件，它只会在你选定的位置上生成一个文件的镜像，不会占用磁盘空间，硬链接 ln 源文件 目标文件，没有参数-s，它会在你选定的位置上生成一个和源文件大小相同的文件，无论是软链接还是硬链接，文件都保持同步变化。
- 第三：ln指令用在链接文件或目录，如同时指定两个以上的文件或目录，且最后的目的地是一个已经存在的目录，则会把前面指定的所有文件或目录复制到该目录中。若同时指定多个文件或目录，且最后的目的地并非是一个已存在的目录，则会出现错误信息。

常用参数:

```
-b 删除，覆盖以前建立的链接  
-s 软链接（符号链接）  
-v 显示详细处理过程
```

实例:

(1) 给文件创建软链接，并显示操作信息

```
ln -sv source.log link.log
```

(2) 给文件创建硬链接，并显示操作信息

```
ln -v source.log link1.log
```

(3) 给目录创建软链接

```
ln -sv /opt/soft/test/test3 /opt/soft/test/test5
```

24、date 命令

显示或设定系统的日期与时间。

命令参数:

```
-d<字符串> 显示字符串所指的日期与时间。字符串前后必须加上双引号。  
-s<字符串> 根据字符串来设置日期与时间。字符串前后必须加上双引号。  
-u 显示GMT。  
%H 小时(00-23)  
%I 小时(00-12)
```

```
%M 分钟(以00-59来表示)
%s 总秒数。起算时间为1970-01-01 00:00:00 UTC。
%S 秒(以本地的惯用法来表示)
%a 星期的缩写。
%A 星期的完整名称。
%d 日期(以01-31来表示)。
%D 日期(含年月日)。
%m 月份(以01-12来表示)。
%y 年份(以00-99来表示)。
%Y 年份(以四位数来表示)。
```

实例：

(1) 显示下一天

```
date +%Y%m%d --date="+1 day" //显示下一天的日期
```

(2) -d参数使用

```
date -d "nov 22" 今年的 11 月 22 日是星期三
date -d '2 weeks' 2周后的日期
date -d 'next monday' (下周一的日期)
date -d next-day +%Y%m%d (明天的日期) 或者: date -d tomorrow +%Y%m%d
date -d last-day +%Y%m%d(昨天的日期) 或者: date -d yesterday +%Y%m%d
date -d last-month +%Y%m(上个月是几月)
date -d next-month +%Y%m(下个月是几月)
```

25、cal 命令

可以用户显示公历（阳历）日历如只有一个参数，则表示年份(1-9999)，如有两个参数，则表示月份和年份：

常用参数：

```
-3 显示前一月，当前月，后一月三个月的日历
-m 显示星期一为第一列
-j 显示在当前年第几天
-y [year]显示当前年[year]份的日历
```

实例：

(1) 显示指定年月日期

```
cal 9 2012
```

(2) 显示2013年每个月日历

```
cal -y 2013
```

(3) 将星期一做为第一列,显示前中后三月

```
cal -3m
```

26、grep 命令

强大的文本搜索命令，grep(Global Regular Expression Print) 全局正则表达式搜索。

grep 的工作方式是这样的，它在一个或多个文件中搜索字符串模板。如果模板包括空格，则必须被引用，模板后的所有字符串被看作文件名。搜索的结果被送到标准输出，不影响原文件内容。

命令格式：

```
grep [option] pattern file|dir
```

常用参数：

```
-A n --after-context 显示匹配字符后n行  
-B n --before-context 显示匹配字符前n行  
-C n --context 显示匹配字符前后n行  
-c --count 计算符合样式的列数  
-i 忽略大小写  
-l 只列出文件内容符合指定的样式的文件名称  
-f 从文件中读取关键词  
-n 显示匹配内容的所在文件中行数  
-R 递归查找文件夹
```

grep 的规则表达式：

```
^ #锚定行的开始 如: '^grep' 匹配所有以grep开头的行。  
$ #锚定行的结束 如: 'grep$' 匹配所有以grep结尾的行。  
. #匹配一个非换行符的字符 如: 'gr.p' 匹配gr后接一个任意字符, 然后是p。  
* #匹配零个或多个先前字符 如: '*grep' 匹配所有一个或多个空格后紧跟grep的行。  
. * #一起用代表任意字符。  
[ ] #匹配一个指定范围内的字符, 如'[Gg]rep' 匹配Grep和grep。  
[^ ] #匹配一个不在指定范围内的字符, 如: '[^A-FH-Z]rep' 匹配不包含A-R和T-Z的一个字母开头, 紧跟rep的行。  
\(.) #标记匹配字符, 如'\\(love\\)', love被标记为1。  
\< #锚定单词的开始, 如: '\\<grep' 匹配包含以grep开头的单词的行。  
\> #锚定单词的结束, 如'grep\\>' 匹配包含以grep结尾的单词的行。  
x\{m\} #重复字符x, m次, 如: '0\{5\}' 匹配包含5个o的行。  
x\{m,\} #重复字符x, 至少m次, 如: 'o\{5,\}' 匹配至少有5个o的行。  
x\{m,n\} #重复字符x, 至少m次, 不多于n次, 如: 'o\{5,10\}' 匹配5--10个o的行。  
\w #匹配文字和数字字符, 也就是[A-Za-z0-9], 如: 'G\w*p' 匹配以G后跟零个或多个文字或数字字符, 然后是p。  
\W #\w的反置形式, 匹配一个或多个非单词字符, 如点号句号等。  
\b #单词锁定符, 如: '\\bgrep\\b'只匹配grep。
```

实例：

(1) 查找指定进程

```
ps -ef | grep svn
```

(2) 查找指定进程个数

```
ps -ef | grep svn -c
```

(3) 从文件中读取关键词

```
cat test1.txt | grep -f key.log
```

(4) 从文件夹中递归查找以grep开头的行，并只列出文件

```
grep -lR '^grep' /tmp
```

(5) 查找非x开关的行内容

```
grep '^[^x]' test.txt
```

(6) 显示包含 ed 或者 at 字符的内容行

```
grep -E 'ed|at' test.txt
```

27、wc 命令

wc(word count)功能为统计指定的文件中字节数、字数、行数，并将统计结果输出

命令格式：

```
wc [option] file..
```

命令参数：

```
-c 统计字节数  
-l 统计行数  
-m 统计字符数  
-w 统计词数，一个字被定义为由空白、跳格或换行字符分隔的字符串
```

实例：

(1) 查找文件的 行数 单词数 字节数 文件名

```
wc text.txt
```

结果：

```
7      8      70      test.txt
```

(2) 统计输出结果的行数

```
cat test.txt | wc -l
```

28、ps 命令

ps(process status)，用来查看当前运行的进程状态，一次性查看，如果需要动态连续结果使用 top

linux上进程有5种状态：

- \1. 运行(正在运行或在运行队列中等待)

- \2. 中断(休眠中, 受阻, 在等待某个条件的形成或接受到信号)
- \3. 不可中断(收到信号不唤醒和不可运行, 进程必须等待直到有中断发生)
- \4. 僵死(进程已终止, 但进程描述符存在, 直到父进程调用wait4()系统调用后释放)
- \5. 停止(进程收到SIGSTOP, SIGSTP, SIGTIN, SIGTOU信号后停止运行)

ps 工具标识进程的5种状态码:

```
D 不可中断 uninterruptible sleep (usually IO)
R 运行 runnable (on run queue)
S 中断 sleeping
T 停止 traced or stopped
Z 僵死 a defunct ("zombie") process
```

命令参数:

```
-A 显示所有进程
a 显示所有进程
-a 显示同一终端下所有进程
c 显示进程真实名称
e 显示环境变量
f 显示进程间的关系
r 显示当前终端运行的进程
-aux 显示所有包含其它使用的进程
```

实例:

(1) 显示当前所有进程环境变量及进程间关系

```
ps -ef
```

(2) 显示当前所有进程

```
ps -A
```

(3) 与grep联用查找某进程

```
ps -aux | grep apache
```

(4) 找出与 cron 与 syslog 这两个服务有关的 PID 号码

```
ps aux | grep '(cron|syslog)'
```

29、top 命令

显示当前系统正在执行的进程的相关信息，包括进程 ID、内存占用率、CPU 占用率等

常用参数:

```
-c 显示完整的进程命令  
-s 保密模式  
-p <进程号> 指定进程显示  
-n <次数>循环显示次数
```

实例：

(1)

```
top - 14:06:23 up 70 days, 16:44, 2 users, load average: 1.25, 1.32, 1.35  
Tasks: 206 total, 1 running, 205 sleeping, 0 stopped, 0 zombie  
Cpu(s): 5.9%us, 3.4%sy, 0.0%ni, 90.4%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st  
Mem: 32949016k total, 14411180k used, 18537836k free, 169884k buffers  
Swap: 32764556k total, 0k used, 32764556k free, 3612636k cached  
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND  
28894 root 22 0 1501m 405m 10m S 52.2 1.3 2534:16 java
```

前五行是当前系统情况整体的统计信息区。

第一行，任务队列信息，同 uptime 命令的执行结果，具体参数说明情况如下：

14:06:23 – 当前系统时间

up 70 days, 16:44 – 系统已经运行了70天16小时44分钟（在这期间系统没有重启过的吆！）

2 users – 当前有2个用户登录系统

load average: 1.15, 1.42, 1.44 – load average后面的三个数分别是1分钟、5分钟、15分钟的负载情况。

load average数据是每隔5秒钟检查一次活跃的进程数，然后按特定算法计算出的数值。如果这个数除以逻辑CPU的数量，结果高于5的时候就表明系统在超负荷运转了。

第二行，Tasks – 任务（进程），具体信息说明如下：

系统现在共有206个进程，其中处于运行中的有1个，205个在休眠（sleep），stoped状态的有0个，zombie状态（僵尸）的有0个。

第三行，cpu状态信息，具体属性说明如下：

```
5.9%us – 用户空间占用CPU的百分比。  
3.4% sy – 内核空间占用CPU的百分比。  
0.0% ni – 改变过优先级的进程占用CPU的百分比  
90.4% id – 空闲CPU百分比  
0.0% wa – IO等待占用CPU的百分比  
0.0% hi – 硬中断（Hardware IRQ）占用CPU的百分比  
0.2% si – 软中断（Software Interrupts）占用CPU的百分比
```

备注：在这里CPU的使用比率和windows概念不同，需要理解linux系统用户空间和内核空间的相关知识！

第四行，内存状态，具体信息如下：

```
32949016k total – 物理内存总量 (32GB)  
14411180k used – 使用中的内存总量 (14GB)  
18537836k free – 空闲内存总量 (18GB)  
169884k buffers – 缓存的内存量 (169M)
```

第五行，swap交换分区信息，具体信息说明如下：

```
32764556k total - 交换区总量 (32GB)
0k used - 使用的交换区总量 (0K)
32764556k free - 空闲交换区总量 (32GB)
3612636k cached - 缓冲的交换区总量 (3.6GB)
```

第六行，空行。

第七行以下：各进程（任务）的状态监控，项目列信息说明如下：

```
PID - 进程id
USER - 进程所有者
PR - 进程优先级
NI - nice值。负值表示高优先级，正值表示低优先级
VIRT - 进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES
RES - 进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA
SHR - 共享内存大小，单位kb
S - 进程状态。D=不可中断的睡眠状态 R=运行 S=睡眠 T=跟踪/停止 Z=僵尸进程
%CPU - 上次更新到现在的CPU时间占用百分比
%MEM - 进程使用的物理内存百分比
TIME+ - 进程使用的CPU时间总计，单位1/100秒
COMMAND - 进程名称（命令名/命令行）
```

top 交互命令

```
h 显示top交互命令帮助信息
c 切换显示命令名称和完整命令行
m 以内存使用率排序
P 根据CPU使用百分比大小进行排序
T 根据时间/累计时间进行排序
W 将当前设置写入~/.toprc文件中
o或者O 改变显示项目的顺序
```

30、kill 命令

发送指定的信号到相应进程。不指定型号将发送SIGTERM (15) 终止指定进程。如果任无法终止该程序可用"-KILL"参数，其发送的信号为SIGKILL(9)，将强制结束进程，使用ps命令或者jobs命令可以查看进程号。root用户将影响用户的进程，非root用户只能影响自己的进程。

常用参数：

```
-l 信号，如果不加信号的编号参数，则使用"-l"参数会列出全部的信号名称
-a 当处理当前进程时，不限制命令名和进程号的对应关系
-p 指定kill 命令只打印相关进程的进程号，而不发送任何信号
-s 指定发送信号
-u 指定用户
```

实例：

(1) 先使用ps查找进程pro1，然后用kill杀掉

```
kill -9 $(ps -ef | grep pro1)
```

31、free 命令

显示系统内存使用情况，包括物理内存、交互区内存(swap)和内核缓冲区内存。

命令参数：

- b 以Byte显示内存使用情况
- k 以kb为单位显示内存使用情况
- m 以mb为单位显示内存使用情况
- g 以gb为单位显示内存使用情况
- s<间隔秒数> 持续显示内存
- t 显示内存使用总合

实例：

(1) 显示内存使用情况

```
free  
free -k  
free -m
```

(2) 以总和的形式显示内存的使用信息

```
free -t
```

(3) 周期性查询内存使用情况

```
free -s 10
```