**Linux 系统启动过程**

linux启动时我们会看到许多启动信息。

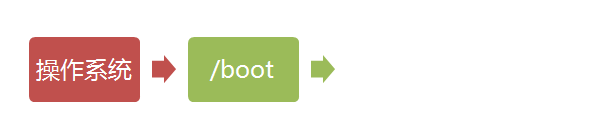
Linux系统的启动过程并不是大家想象中的那么复杂，其过程可以分为5个阶段：

* 内核的引导。
* 运行init。
* 系统初始化。
* 建立终端 。
* 用户登录系统。

**内核引导**

当计算机打开电源后，首先是BIOS开机自检，按照BIOS中设置的启动设备（通常是硬盘）来启动。

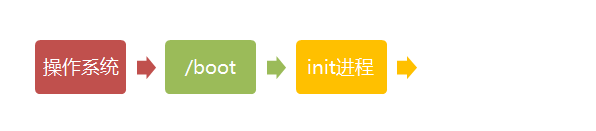
操作系统接管硬件以后，首先读入 /boot 目录下的内核文件。



**运行init**

init 进程是系统所有进程的起点，你可以把它比拟成系统所有进程的老祖宗，没有这个进程，系统中任何进程都不会启动。

init 程序首先是需要读取配置文件 /etc/inittab。



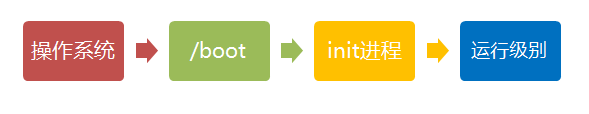
**运行级别**

许多程序需要开机启动。它们在Windows叫做"服务"（service），在Linux就叫做"守护进程"（daemon）。

init进程的一大任务，就是去运行这些开机启动的程序。

但是，不同的场合需要启动不同的程序，比如用作服务器时，需要启动Apache，用作桌面就不需要。

Linux允许为不同的场合，分配不同的开机启动程序，这就叫做"运行级别"（runlevel）。也就是说，启动时根据"运行级别"，确定要运行哪些程序。



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

**系统初始化**

在init的配置文件中有这么一行： si::sysinit:/etc/rc.d/rc.sysinit　它调用执行了/etc/rc.d/rc.sysinit，而rc.sysinit是一个bash shell的脚本，它主要是完成一些系统初始化的工作，rc.sysinit是每一个运行级别都要首先运行的重要脚本。

它主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。

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

这一行表示以5为参数运行/etc/rc.d/rc，/etc/rc.d/rc是一个Shell脚本，它接受5作为参数，去执行/etc/rc.d/rc5.d/目录下的所有的rc启动脚本，/etc/rc.d/rc5.d/目录中的这些启动脚本实际上都是一些连接文件，而不是真正的rc启动脚本，真正的rc启动脚本实际上都是放在/etc/rc.d/init.d/目录下。

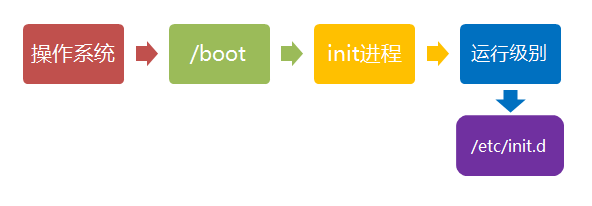
而这些rc启动脚本有着类似的用法，它们一般能接受start、stop、restart、status等参数。

/etc/rc.d/rc5.d/中的rc启动脚本通常是K或S开头的连接文件，对于以以S开头的启动脚本，将以start参数来运行。

而如果发现存在相应的脚本也存在K打头的连接，而且已经处于运行态了(以/var/lock/subsys/下的文件作为标志)，则将首先以stop为参数停止这些已经启动了的守护进程，然后再重新运行。

这样做是为了保证是当init改变运行级别时，所有相关的守护进程都将重启。

至于在每个运行级中将运行哪些守护进程，用户可以通过chkconfig或setup中的"System Services"来自行设定。



**建立终端**

rc执行完毕后，返回init。这时基本系统环境已经设置好了，各种守护进程也已经启动了。

init接下来会打开6个终端，以便用户登录系统。在inittab中的以下6行就是定义了6个终端：

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

4:2345:respawn:/sbin/mingetty tty4

5:2345:respawn:/sbin/mingetty tty5

6:2345:respawn:/sbin/mingetty tty6

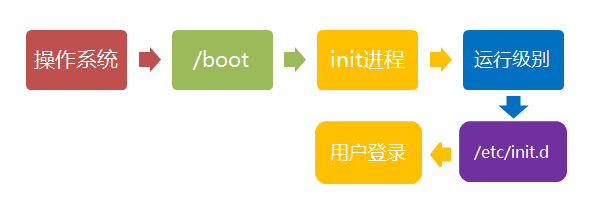
从上面可以看出在2、3、4、5的运行级别中都将以respawn方式运行mingetty程序，mingetty程序能打开终端、设置模式。

同时它会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，而用户输入的用户将作为参数传给login程序来验证用户的身份。

**用户登录系统**

一般来说，用户的登录方式有三种：

* （1）命令行登录
* （2）ssh登录
* （3）图形界面登录



对于运行级别为5的图形方式用户来说，他们的登录是通过一个图形化的登录界面。登录成功后可以直接进入KDE、Gnome等窗口管理器。

而本文主要讲的还是文本方式登录的情况：当我们看到mingetty的登录界面时，我们就可以输入用户名和密码来登录系统了。

Linux的账号验证程序是login，login会接收mingetty传来的用户名作为用户名参数。

然后login会对用户名进行分析：如果用户名不是root，且存在/etc/nologin文件，login将输出nologin文件的内容，然后退出。

这通常用来系统维护时防止非root用户登录。只有/etc/securetty中登记了的终端才允许root用户登录，如果不存在这个文件，则root可以在任何终端上登录。

/etc/usertty文件用于对用户作出附加访问限制，如果不存在这个文件，则没有其他限制。

<="" p="" style="color: rgb(0, 0, 0); font-family: 'Microsoft Yahei', 'Helvetica Neue', Helvetica, Arial, sans-serif; font-size: 12px; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: normal; orphans: auto; text-align: start; text-indent: 0px; text-transform: none; white-space: normal; widows: 1; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255);">

**图形模式与文字模式的切换方式**

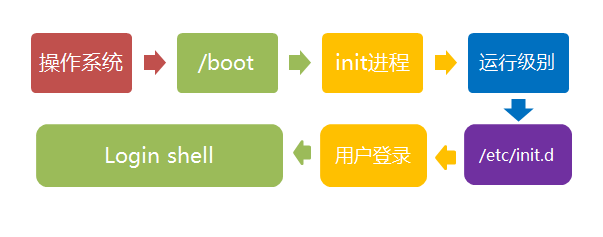
Linux预设提供了六个命令窗口终端机让我们来登录。

默认我们登录的就是第一个窗口，也就是tty1，这个六个窗口分别为tty1,tty2 … tty6，你可以按下Ctrl + Alt + F1 ~ F6 来切换它们。

如果你安装了图形界面，默认情况下是进入图形界面的，此时你就可以按Ctrl + Alt + F1 ~ F6来进入其中一个命令窗口界面。

当你进入命令窗口界面后再返回图形界面只要按下Ctrl + Alt + F7 就回来了。

如果你用的vmware 虚拟机，命令窗口切换的快捷键为 Alt + Space + F1~F6. 如果你在图形界面下请按Alt + Shift + Ctrl + F1~F6 切换至命令窗口。



**Linux 关机**

在linux领域内大多用在服务器上，很少遇到关机的操作。毕竟服务器上跑一个服务是永无止境的，除非特殊情况下，不得已才会关机。

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

关机指令为：shutdown ，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：

sync 将数据由内存同步到硬盘中。

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

最后总结一下，不管是重启系统还是关闭系统，首先要运行sync命令，把内存中的数据写到磁盘中。

关机的命令有 shutdown –h now halt poweroff 和 init 0 , 重启系统的命令有 shutdown –r now reboot init 6.

**Linux 系统目录结构**

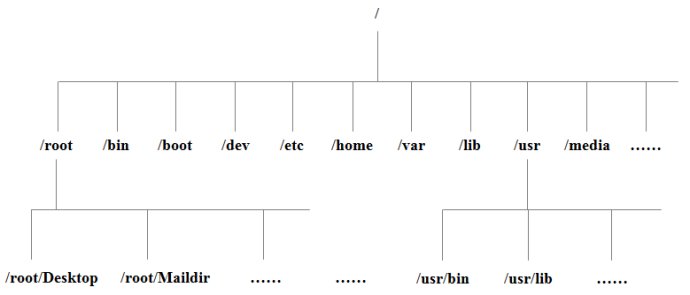
登录系统后，在当前命令窗口下输入命令：

 ls /

你会看到如下图所示:

[4_20](http://www.w3cschool.cc/wp-content/uploads/2014/06/4_20.png)

树状目录结构：



以下是对这些目录的解释：

* **/bin**：  
  bin是Binary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：**  
  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**：  
  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**：  
   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**：  
  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，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种被创建。

* **/tmp**：  
  这个目录是用来存放一些临时文件的。
* **/usr**：  
   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，类似与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**：  
  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。

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。

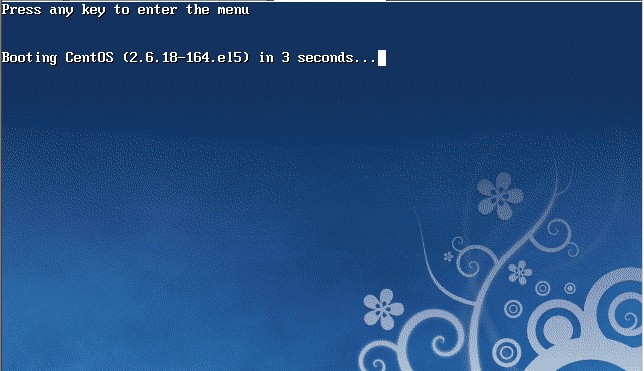
/var： 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在/var/log 目录下，另外mail的预设放置也是在这里。

**Linux 忘记密码解决方法**

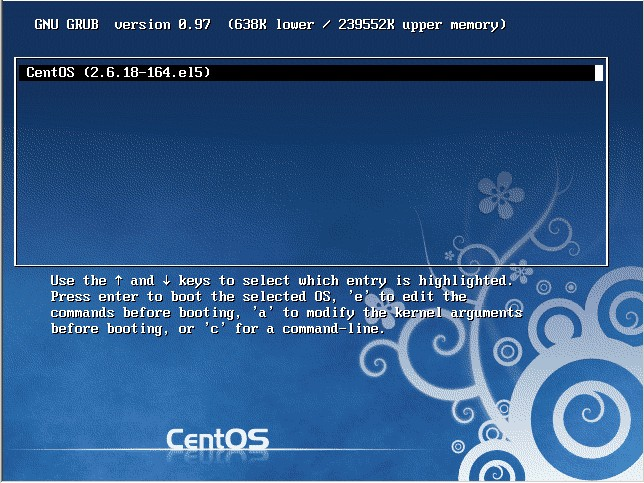
很多朋友经常会忘记Linux系统的root密码，linux系统忘记root密码的情况该怎么办呢？重新安装系统吗？当然不用！进入单用户模式更改一下root密码即可。

步骤如下：

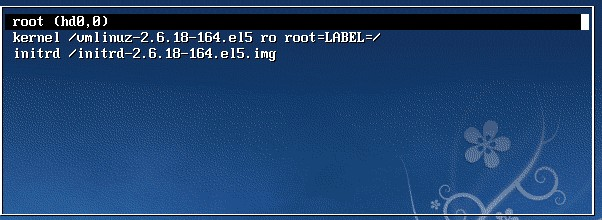
**重启linux系统**



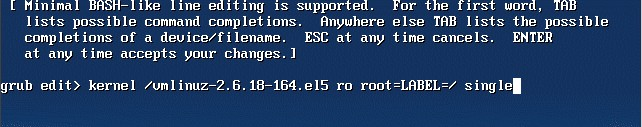
3 秒之内要按一下回车，出现如下界面



然后输入e

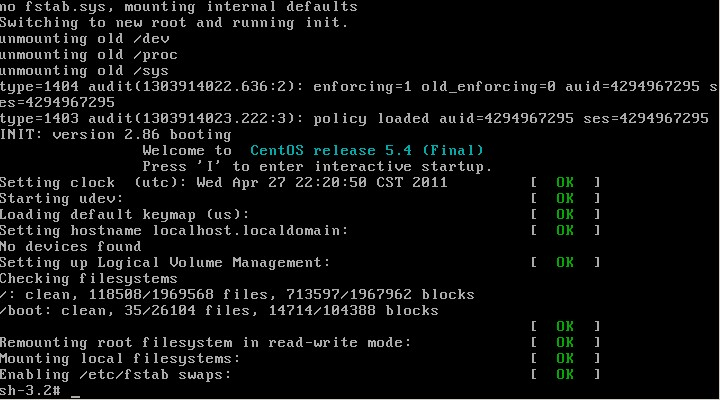


在 第二行最后边输入 single，有一个空格。具体方法为按向下尖头移动到第二行，按"e"进入编辑模式

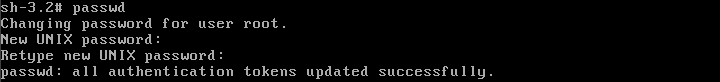
在后边加上single 回车

4_25

最后按"b"启动，启动后就进入了单用户模式了



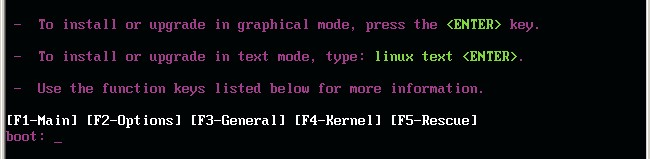
此时已经进入到单用户模式了，你可以更改root密码了。更密码的命令为 passwd



【**使用系统安装光盘的救援模式**】

救援模式即rescue ，这个模式主要是应用于，系统无法进入的情况。如，grub损坏或者某一个配置文件修改出错。如何使用rescue模式呢？

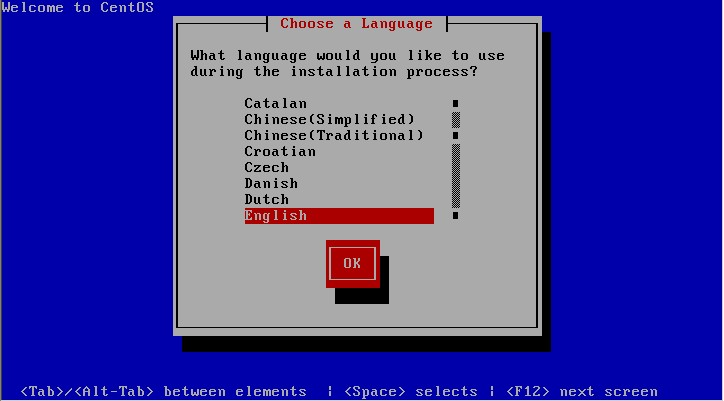
光盘启动，按F5 进入rescue模式



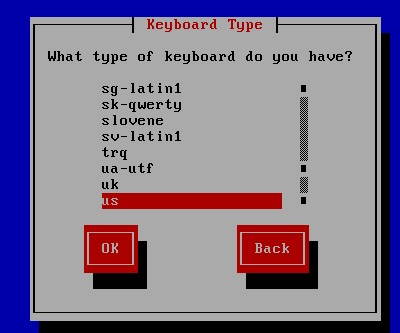
输入linux rescue 回车

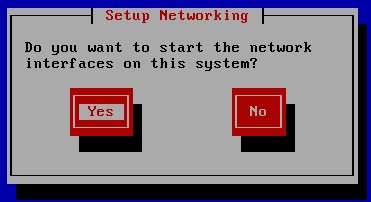


选择语言，笔者建议你选择英语

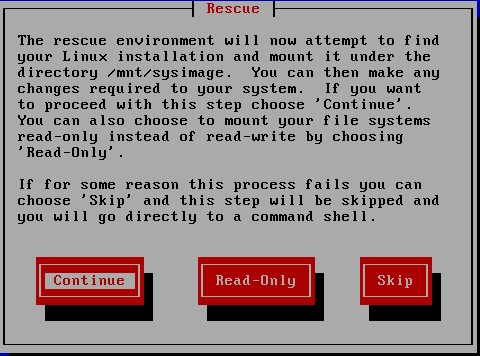


选择us 键盘





这里问你是否启动网络，有时候可能会联网调试。我们选no

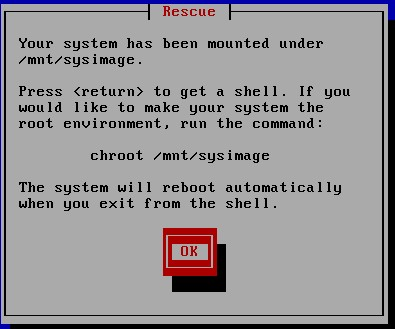


这里告诉我们，接下来会把系统挂载在/mnt/sysimage 中。

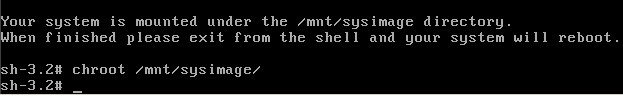
其中有三个选项:

* Continue 就是挂载后继续下一步。
* Read-Only 挂载成只读，这样更安全，有时文件系统损坏时，只读模式会防止文件系统近一步损坏。
* Skip就是不挂载，进入一个命令窗口模式。

这里我们选择Continue。



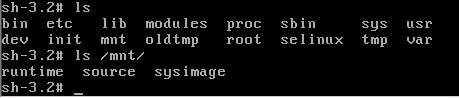
至此，系统已经挂载到了/mnt/sysimage中。接下来回车，输入chroot /mnt/sysimage 进入管理员环境。



**提示：** 其实也可以到rescue模式下更改root的密码的。这个rescue模式和windows PE系统很相近。

当运行了chroot /mnt/sysimage/ 后，再ls 看到目录结构和原来系统中的目录结构是一样的。

没错！现在的环境和原来系统的环境是一模一样的。你可以输入exit 或者按Ctrl + D退出这个环境。然后你再ls 看一下



这个目录其实就是rescue模式下的目录结构，而我们的系统文件全部在 /mnt/sysimage目录下。

# Linux 远程登录

Linux一般作为服务器使用，而服务器一般放在机房，你不可能在机房操作你的Linux服务器。

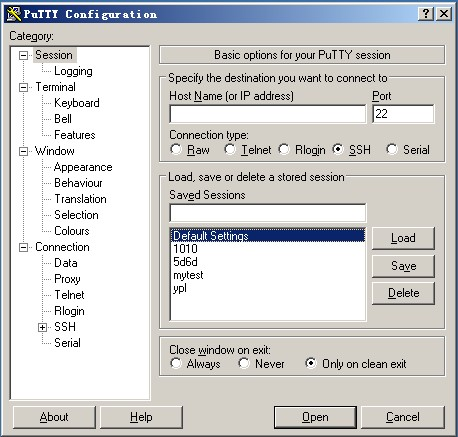
这事我们就需要远程登录到Linux服务器来管理维护系统。

Linux系统中是通过ssh服务实现的远程登录功能，默认ssh服务端口号为 22。

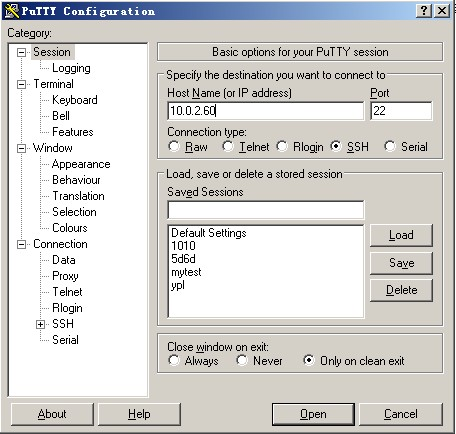
Window系统上 Linux 远程登录客户端有SecureCRT, Putty, SSH Secure Shell等，本文以Putty为例来登录远程服务器。

putty下载地址：<http://www.putty.org/>

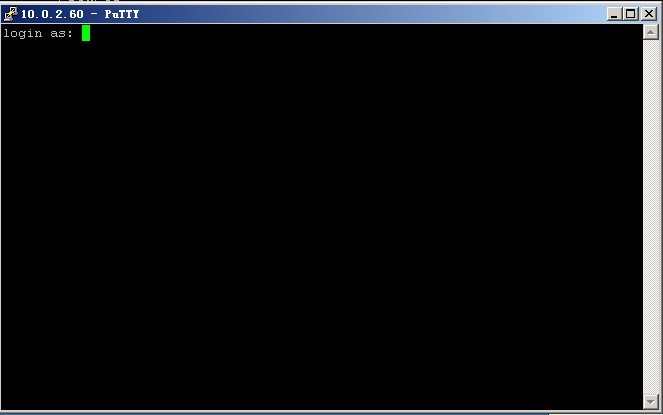
如果你下载了putty，请双击putty.exe 然后弹出如下的窗口。



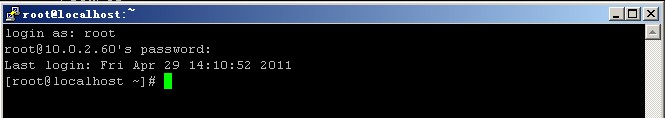
在Host Name( or IP address) 下面的框中输入你要登录的远程服务器IP(可以通过ifconfig命令查看服务器ip)，然后回车。



此时，提示我们输入要登录的用户名。



输入root 然后回车，再输入密码，就能登录到远程的linux系统了。

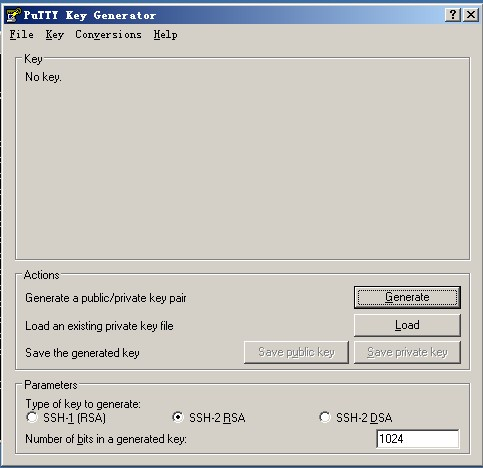


## 使用密钥认证机制远程登录linux

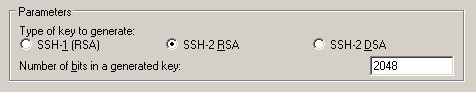
SSH 为 Secure Shell 的缩写，由 IETF 的网络工作小组（Network Working Group）所制定。

SSH 为建立在应用层和传输层基础上的安全协议。

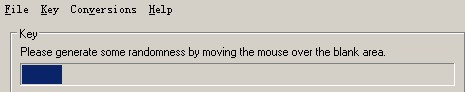
首先使用工具 PUTTYGEN.EXE 生成密钥对。打开工具PUTTYGEN.EXE后如下图所示：



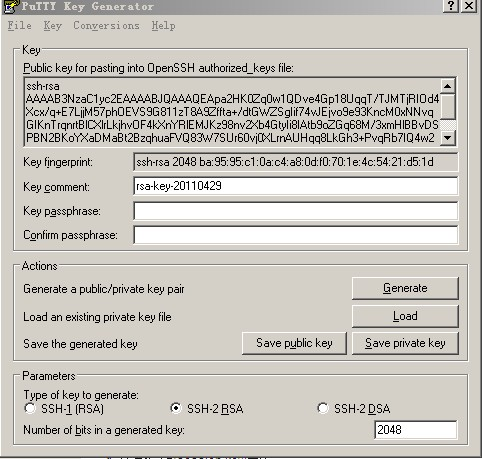
该工具可以生成三种格式的key ：SSH-1(RSA) SSH-2(RSA) SSH-2(DSA) ，我们采用默认的格式即SSH-2(RSA)。Number of bits in a generated key 这个是指生成的key的大小，这个数值越大，生成的key就越复杂，安全性就越高。这里我们写2048.



然后单击Generate 开始生成密钥对：



注意的是，在这个过程中鼠标要来回的动，否则这个进度条是不会动的。



到这里，密钥对已经生成了。你可以给你的密钥输入一个密码，（在Key Passphrase那里）也可以留空。然后点 Save public key 保存公钥，点 Save private Key 保存私钥。笔者建议你放到一个比较安全的地方，一来防止别人偷窥，二来防止误删除。接下来就该到远程linux主机上设置了。

1）创建目录 /root/.ssh 并设置权限

[root@localhost ~]# mkdir /root/.ssh mkdir 命令用来创建目录，以后会详细介绍，暂时只了解即可。

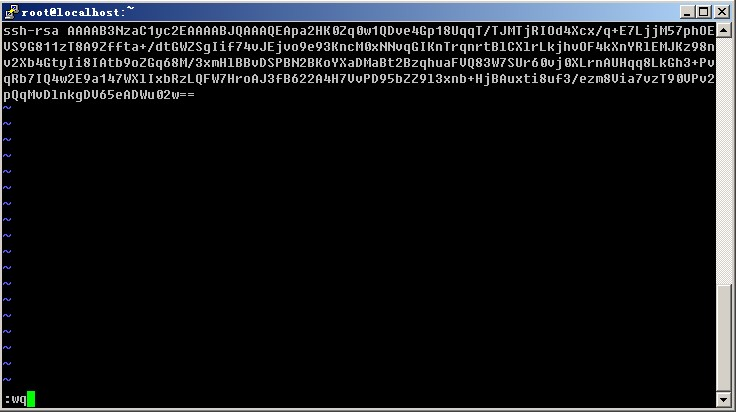
[root@localhost ~]# chmod 700 /root/.ssh chmod 命令是用来修改文件属性权限的，以后会详细介绍。

2）创建文件 / root/.ssh/authorized\_keys

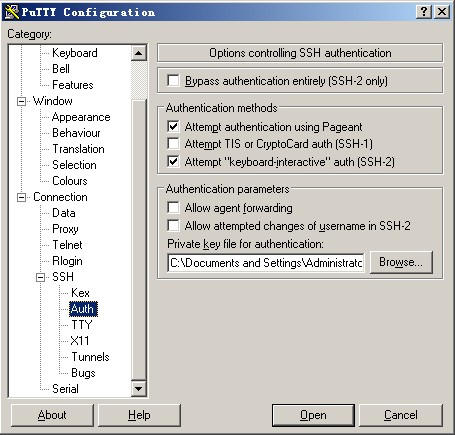
[root@localhost ~]# vim /root/.ssh/authorized\_keys vim 命令是编辑一个文本文件的命令，同样在后续章节详细介绍。

3）打开刚才生成的public key 文件，建议使用写字板打开，这样看着舒服一些，复制从AAAA开头至 "---- END SSH2 PUBLIC KEY ----" 该行上的所有内容，粘贴到/root/.ssh/authorized\_keys 文件中，要保证所有字符在一行。（可以先把复制的内容拷贝至记事本，然后编辑成一行载粘贴到该文件中）。

在这里要简单介绍一下，如何粘贴，用vim打开那个文件后，该文件不存在，所以vim会自动创建。按一下字母"i"然后同时按shift + Insert 进行粘贴（或者单击鼠标邮件即可），前提是已经复制到剪切板中了。粘贴好后，然后把光标移动到该行最前面输入ssh-ras ，然后按空格。再按ESC，然后输入冒号wq 即 :wq 就保存了。格式如下图：



4）再设置putty选项，点窗口左侧的SSh –> Auth ，单击窗口右侧的Browse… 选择刚刚生成的私钥， 再点Open ，此时输入root，就不用输入密码就能登录了。



如果在前面你设置了Key Passphrase ，那么此时就会提示你输入密码的。为了更加安全建议大家要设置一个Key Passphrase。

# Linux 文件基本属性

Linux系统是一种典型的多用户系统，不同的用户处于不同的地位，拥有不同的权限。为了保护系统的安全性，Linux系统对不同的用户访问同一文件（包括目录文件）的权限做了不同的规定。

在Linux中我们可以使用ll或者ls –l命令来显示一个文件的属性以及文件所属的用户和组，如：

[root@www /]# ls -l

total 64

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

dr-xr-xr-x 4 root root 4096 Apr 19 2012 boot

……

实例中，bin文件的第一个属性用"d"表示。"d"在Linux中代表该文件是一个目录文件。

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。

每个文件的属性由左边第一部分的10个字符来确定（如下图）。



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。

## Linux文件属主和属组

[root@www /]# ls -l

total 64

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

dr-xr-xr-x 4 root root 4096 Apr 19 2012 boot

……

对于文件来说，它都有一个特定的所有者，也就是对该文件具有所有权的用户。

同时，在Linux系统中，用户是按组分类的，一个用户属于一个或多个组。

文件所有者以外的用户又可以分为文件所有者的同组用户和其他用户。

因此，Linux系统按文件所有者、文件所有者同组用户和其他用户来规定了不同的文件访问权限。

在以上实例中，bin文件是一个目录文件，属主和属组都为root，属主有可读、可写、可执行的权限；与属主同组的其他用户有可读和可执行的权限；其他用户也有可读和可执行的权限。

### 更改文件属性

#### 1、chgrp：更改文件属组

语法：

chgrp [-R] 属组名文件名

参数选项

* -R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上-R的参数，那么该目录下的所有文件的属组都会更改。

#### 2、chown：更改文件属主，也可以同时更改文件属组

语法：

chown [–R] 属主名 文件名

chown [-R] 属主名：属组名 文件名

进入 /root 目录（~）将install.log的拥有者改为bin这个账号：

[root@www ~] cd ~

[root@www ~]# chown bin install.log

[root@www ~]# ls -l

-rw-r--r-- 1 bin users 68495 Jun 25 08:53 install.log

将install.log的拥有者与群组改回为root：

[root@www ~]# chown root:root install.log

[root@www ~]# ls -l

-rw-r--r-- 1 root root 68495 Jun 25 08:53 install.log

#### 3、chmod：更改文件9个属性

Linux文件属性有两种设置方法，一种是数字，一种是符号。

Linux文件的基本权限就有九个，分别是owner/group/others三种身份各有自己的read/write/execute权限。

先复习一下刚刚上面提到的数据：文件的权限字符为：『-rwxrwxrwx』， 这九个权限是三个三个一组的！其中，我们可以使用数字来代表各个权限，各权限的分数对照表如下：

* r:4
* w:2
* x:1

每种身份(owner/group/others)各自的三个权限(r/w/x)分数是需要累加的，例如当权限为： [-rwxrwx---] 分数则是：

* owner = rwx = 4+2+1 = 7
* group = rwx = 4+2+1 = 7
* others= --- = 0+0+0 = 0

所以等一下我们设定权限的变更时，该文件的权限数字就是770啦！变更权限的指令chmod的语法是这样的：

chmod [-R] xyz 文件或目录

选项与参数：

* xyz : 就是刚刚提到的数字类型的权限属性，为 rwx 属性数值的相加。
* -R : 进行递归(recursive)的持续变更，亦即连同次目录下的所有文件都会变更

举例来说，如果要将.bashrc这个文件所有的权限都设定启用，那么命令如下：

[root@www ~]# ls -al .bashrc

-rw-r--r-- 1 root root 395 Jul 4 11:45 .bashrc

[root@www ~]# chmod 777 .bashrc

[root@www ~]# ls -al .bashrc

-rwxrwxrwx 1 root root 395 Jul 4 11:45 .bashrc

那如果要将权限变成 -rwxr-xr-- 呢？那么权限的分数就成为 [4+2+1][4+0+1][4+0+0]=754。

#### 符号类型改变文件权限

还有一个改变权限的方法呦！从之前的介绍中我们可以发现，基本上就九个权限分别是(1)user (2)group (3)others三种身份啦！ 那么我们就可以藉由u, g, o来代表三种身份的权限！

此外， a 则代表 all 亦即全部的身份！那么读写的权限就可以写成r, w, x！也就是可以使用底下的方式来看：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| chmod | u g o a | +(加入) -(除去) =(设定) | r w x | 文件或目录 |

如果我们需要将文件权限设置为 -rwxr-xr-- ，可以使用 chmod u=rwx,g=rx,o=r 文件名 来设定:

[root@www ~]# ls -al .bashrc

-rwxr-xr-x 1 root root 395 Jul 4 11:45 .bashrc

[root@www ~]# chmod a+w .bashrc

[root@www ~]# ls -al .bashrc

-rwxrwxrwx 1 root root 395 Jul 4 11:45 .bashrc

而如果是要将权限去掉而不改变其他已存在的权限呢？例如要拿掉全部人的可执行权限，则：

[root@www ~]# chmod a-x .bashrc

[root@www ~]# ls -al .bashrc

-rw-rw-rw- 1 root root 395 Jul 4 11:45 .bashrc

**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啦！

**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: 列出目录
* cd：切换目录
* pwd：显示目前的目录
* mkdir：创建一个新的目录
* rmdir：删除一个空的目录
* cp: 复制文件或目录
* rm: 移除文件或目录

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，连同隐藏档( 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w3cschool.cc目录

[root@www ~]# mkdir w3cschool.cc

#使用绝对路径切换到w3cschool.cc目录

[root@www ~]# cd /root/w3cschool.cc/

#使用相对路径切换到w3cschool.cc目录

[root@www ~]# cd ./w3cschool.cc/

# 表示回到自己的家目录，亦即是 /root 这个目录

[root@www w3cschool.cc]# 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。

范例：单纯显示出目前的工作目录：

[root@www ~]# pwd

/root <== 显示出目录啦～

范例：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)递回创建起来！

范例：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！

范例：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。

**rmdir (删除空的目录)**

语法：

rmdir [-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 w3cschool.cc 目录

[root@www tmp]# rmdir w3cschool.cc/

范例：将於mkdir范例中创建的目录(/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]# rmdir test <==可直接删除掉，没问题

[root@www tmp]# rmdir test1 <==因为尚有内容，所以无法删除！

rmdir: `test1': Directory not empty

[root@www tmp]# rmdir -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 rmdir 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 [-adfilprsu] 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 -pdr 的意思，至於 pdr 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 ：递回持续复制，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，将家目录下的 .bashrc 复制到 /tmp 下，并更名为 bashr

[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 ：递回删除啊！最常用在目录的删除了！这是非常危险的选项！！！

将刚刚在 cp 的范例中创建的 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 的倒著写！
* 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 ：行号在萤幕的最左方显示；  
  -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。

范例一：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.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

默认的情况中，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

# 默认的情况中，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

# Linux 用户和用户组管理

Linux系统是一个多用户多任务的分时操作系统，任何一个要使用系统资源的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。

用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性保护。

每个用户账号都拥有一个惟一的用户名和各自的口令。

用户在登录时键入正确的用户名和口令后，就能够进入系统和自己的主目录。

实现用户账号的管理，要完成的工作主要有如下几个方面：

* 用户账号的添加、删除与修改。
* 用户口令的管理。
* 用户组的管理。

## 一、Linux系统用户账号的管理

用户账号的管理工作主要涉及到用户账号的添加、修改和删除。

添加用户账号就是在系统中创建一个新账号，然后为新账号分配用户号、用户组、主目录和登录Shell等资源。刚添加的账号是被锁定的，无法使用。

### 1、添加新的用户账号使用useradd命令，其语法如下：

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 /usr/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/usr/sam（/usr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。

### 3、删除帐号

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 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。

### 4、修改帐号

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。

### 5、用户口令的管理

用户管理的一项重要内容是用户口令的管理。用户账号刚创建时没有口令，但是被系统锁定，无法使用，必须为其指定口令后才可以使用，即使是指定空口令。

指定和修改用户口令的Shell命令是passwd。超级用户可以为自己和其他用户指定口令，普通用户只能用它修改自己的口令。命令的格式为：

passwd 选项 用户名

可使用的选项：

* -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。

如果默认用户名，则修改当前用户的口令。

例如，假设当前用户是sam，则下面的命令修改该用户自己的口令：

$ 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口令：

# passwd sam

New password:\*\*\*\*\*\*\*

Re-enter new password:\*\*\*\*\*\*\*

普通用户修改自己的口令时，passwd命令会先询问原口令，验证后再要求用户输入两遍新口令，如果两次输入的口令一致，则将这个口令指定给用户；而超级用户为用户指定口令时，就不需要知道原口令。

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

为用户指定空口令时，执行下列形式的命令：

# passwd -d sam

此命令将用户sam的口令删除，这样用户sam下一次登录时，系统就不再询问口令。

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

## 二、Linux系统用户组的管理

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。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。

### 1、增加一个新的用户组使用groupadd命令。其格式如下：

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。

### 2、如果要删除一个已有的用户组，使用groupdel命令，其格式如下：

groupdel 用户组

#### 例如：

# groupdel group1

此命令从系统中删除组group1。

### 3、修改用户组的属性使用groupmod命令。其语法如下：

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。

### 4、如果一个用户同时属于多个用户组，那么用户可以在用户组之间切换，以便具有其他用户组的权限。

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。

## 三、与用户账号有关的系统文件

完成用户管理的工作有许多种方法，但是每一种方法实际上都是对有关的系统文件进行修改。

与用户和用户组相关的信息都存放在一些系统文件中，这些文件包括/etc/passwd, /etc/shadow, /etc/group等。

下面分别介绍这些文件的内容。

### 1、/etc/passwd文件是用户管理工作涉及的最重要的一个文件。

Linux系统中的每个用户都在/etc/passwd文件中有一个对应的记录行，它记录了这个用户的一些基本属性。

这个文件对所有用户都是可读的。它的内容类似下面的例子：

＃ cat /etc/passwd

root:x:0:0:Superuser:/:

daemon:x:1:1:System daemons:/etc:

bin:x:2:2:Owner of system commands:/bin:

sys:x:3:3:Owner of system files:/usr/sys:

adm:x:4:4:System accounting:/usr/adm:

uucp:x:5:5:UUCP administrator:/usr/lib/uucp:

auth:x:7:21:Authentication administrator:/tcb/files/auth:

cron:x:9:16:Cron daemon:/usr/spool/cron:

listen:x:37:4:Network daemon:/usr/net/nls:

lp:x:71:18:Printer administrator:/usr/spool/lp:

sam:x:200:50:Sam san:/usr/sam:/bin/sh

从上面的例子我们可以看到，/etc/passwd中一行记录对应着一个用户，每行记录又被冒号(:)分隔为7个字段，其格式和具体含义如下：

用户名:口令:用户标识号:组标识号:注释性描述:主目录:登录Shell

### 1）"用户名"是代表用户账号的字符串。

通常长度不超过8个字符，并且由大小写字母和/或数字组成。登录名中不能有冒号(:)，因为冒号在这里是分隔符。

为了兼容起见，登录名中最好不要包含点字符(.)，并且不使用连字符(-)和加号(+)打头。

### 2）“口令”一些系统中，存放着加密后的用户口令字。

虽然这个字段存放的只是用户口令的加密串，不是明文，但是由于/etc/passwd文件对所有用户都可读，所以这仍是一个安全隐患。因此，现在许多Linux 系统（如SVR4）都使用了shadow技术，把真正的加密后的用户口令字存放到/etc/shadow文件中，而在/etc/passwd文件的口令字段中只存放一个特殊的字符，例如“x”或者“\*”。

### 3）“用户标识号”是一个整数，系统内部用它来标识用户。

一般情况下它与用户名是一一对应的。如果几个用户名对应的用户标识号是一样的，系统内部将把它们视为同一个用户，但是它们可以有不同的口令、不同的主目录以及不同的登录Shell等。

通常用户标识号的取值范围是0～65 535。0是超级用户root的标识号，1～99由系统保留，作为管理账号，普通用户的标识号从100开始。在Linux系统中，这个界限是500。

### 4）“组标识号”字段记录的是用户所属的用户组。

它对应着/etc/group文件中的一条记录。

### 5)“注释性描述”字段记录着用户的一些个人情况。

例如用户的真实姓名、电话、地址等，这个字段并没有什么实际的用途。在不同的Linux 系统中，这个字段的格式并没有统一。在许多Linux系统中，这个字段存放的是一段任意的注释性描述文字，用做finger命令的输出。

### 6)“主目录”，也就是用户的起始工作目录。

它是用户在登录到系统之后所处的目录。在大多数系统中，各用户的主目录都被组织在同一个特定的目录下，而用户主目录的名称就是该用户的登录名。各用户对自己的主目录有读、写、执行（搜索）权限，其他用户对此目录的访问权限则根据具体情况设置。

### 7)用户登录后，要启动一个进程，负责将用户的操作传给内核，这个进程是用户登录到系统后运行的命令解释器或某个特定的程序，即Shell。

Shell是用户与Linux系统之间的接口。Linux的Shell有许多种，每种都有不同的特点。常用的有sh(Bourne Shell), csh(C Shell), ksh(Korn Shell), tcsh(TENEX/TOPS-20 type C Shell), bash(Bourne Again Shell)等。

系统管理员可以根据系统情况和用户习惯为用户指定某个Shell。如果不指定Shell，那么系统使用sh为默认的登录Shell，即这个字段的值为/bin/sh。

用户的登录Shell也可以指定为某个特定的程序（此程序不是一个命令解释器）。

利用这一特点，我们可以限制用户只能运行指定的应用程序，在该应用程序运行结束后，用户就自动退出了系统。有些Linux 系统要求只有那些在系统中登记了的程序才能出现在这个字段中。

### 8)系统中有一类用户称为伪用户（psuedo users）。

这些用户在/etc/passwd文件中也占有一条记录，但是不能登录，因为它们的登录Shell为空。它们的存在主要是方便系统管理，满足相应的系统进程对文件属主的要求。

常见的伪用户如下所示：

伪 用 户 含 义

bin 拥有可执行的用户命令文件

sys 拥有系统文件

adm 拥有帐户文件

uucp UUCP使用

lp lp或lpd子系统使用

nobody NFS使用

## 拥有帐户文件

### 1、除了上面列出的伪用户外，还有许多标准的伪用户，例如：audit, cron, mail, usenet等，它们也都各自为相关的进程和文件所需要。

由于/etc/passwd文件是所有用户都可读的，如果用户的密码太简单或规律比较明显的话，一台普通的计算机就能够很容易地将它破解，因此对安全性要求较高的Linux系统都把加密后的口令字分离出来，单独存放在一个文件中，这个文件是/etc/shadow文件。 有超级用户才拥有该文件读权限，这就保证了用户密码的安全性。

### 2、/etc/shadow中的记录行与/etc/passwd中的一一对应，它由pwconv命令根据/etc/passwd中的数据自动产生

它的文件格式与/etc/passwd类似，由若干个字段组成，字段之间用":"隔开。这些字段是：

登录名:加密口令:最后一次修改时间:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志

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，长度为13个字符。如果为空，则对应用户没有口令，登录时不需要口令；如果含有不属于集合 { ./0-9A-Za-z }中的字符，则对应的用户不能登录。
3. "最后一次修改时间"表示的是从某个时刻起，到用户最后一次修改口令时的天数。时间起点对不同的系统可能不一样。例如在SCO Linux 中，这个时间起点是1970年1月1日。
4. "最小时间间隔"指的是两次修改口令之间所需的最小天数。
5. "最大时间间隔"指的是口令保持有效的最大天数。
6. "警告时间"字段表示的是从系统开始警告用户到用户密码正式失效之间的天数。
7. "不活动时间"表示的是用户没有登录活动但账号仍能保持有效的最大天数。
8. "失效时间"字段给出的是一个绝对的天数，如果使用了这个字段，那么就给出相应账号的生存期。期满后，该账号就不再是一个合法的账号，也就不能再用来登录了。

下面是/etc/shadow的一个例子：

＃ cat /etc/shadow

root:Dnakfw28zf38w:8764:0:168:7:::

daemon:\*::0:0::::

bin:\*::0:0::::

sys:\*::0:0::::

adm:\*::0:0::::

uucp:\*::0:0::::

nuucp:\*::0:0::::

auth:\*::0:0::::

cron:\*::0:0::::

listen:\*::0:0::::

lp:\*::0:0::::

sam:EkdiSECLWPdSa:9740:0:0::::

### 3、用户组的所有信息都存放在/etc/group文件中。

将用户分组是Linux 系统中对用户进行管理及控制访问权限的一种手段。

每个用户都属于某个用户组；一个组中可以有多个用户，一个用户也可以属于不同的组。

当一个用户同时是多个组中的成员时，在/etc/passwd文件中记录的是用户所属的主组，也就是登录时所属的默认组，而其他组称为附加组。

用户要访问属于附加组的文件时，必须首先使用newgrp命令使自己成为所要访问的组中的成员。

用户组的所有信息都存放在/etc/group文件中。此文件的格式也类似于/etc/passwd文件，由冒号(:)隔开若干个字段，这些字段有：

组名:口令:组标识号:组内用户列表

1. "组名"是用户组的名称，由字母或数字构成。与/etc/passwd中的登录名一样，组名不应重复。
2. "口令"字段存放的是用户组加密后的口令字。一般Linux 系统的用户组都没有口令，即这个字段一般为空，或者是\*。
3. "组标识号"与用户标识号类似，也是一个整数，被系统内部用来标识组。
4. "组内用户列表"是属于这个组的所有用户的列表/b]，不同用户之间用逗号(,)分隔。这个用户组可能是用户的主组，也可能是附加组。

/etc/group文件的一个例子如下：

root::0:root

bin::2:root,bin

sys::3:root,uucp

adm::4:root,adm

daemon::5:root,daemon

lp::7:root,lp

users::20:root,sam

### 四、添加量用户批

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。

范例文件 passwd.txt 内容如下：

user001:密码

user002:密码

user003:密码

user004:密码

user005:密码

user006:密码

### （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

这样就完成了大量用户的创建了，之后您可以到/home下检查这些用户宿主目录的权限设置是否都正确，并登录验证用户密码是否正确。

**Linux 磁盘管理**

Linux磁盘管理好坏管理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
* du：检查磁盘空间使用量
* fdisk：用于磁盘分区

**df**

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。

语法：

df [-ahikHTm] [目录或文件名]

选项与参数：

* -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

**实例 1**

将系统内所有的文件系统列出来！

[root@www ~]# df

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/hdc2 9920624 3823112 5585444 41% /

/dev/hdc3 4956316 141376 4559108 4% /home

/dev/hdc1 101086 11126 84741 12% /boot

tmpfs 371332 0 371332 0% /dev/shm

在 Linux 底下如果 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！

**实例 2**

将容量结果以易读的容量格式显示出来

[root@www ~]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/hdc2 9.5G 3.7G 5.4G 41% /

/dev/hdc3 4.8G 139M 4.4G 4% /home

/dev/hdc1 99M 11M 83M 12% /boot

tmpfs 363M 0 363M 0% /dev/shm

**实例 3**

将系统内的所有特殊文件格式及名称都列出来

[root@www ~]# df -aT

Filesystem Type 1K-blocks Used Available Use% Mounted on

/dev/hdc2 ext3 9920624 3823112 5585444 41% /

proc proc 0 0 0 - /proc

sysfs sysfs 0 0 0 - /sys

devpts devpts 0 0 0 - /dev/pts

/dev/hdc3 ext3 4956316 141376 4559108 4% /home

/dev/hdc1 ext3 101086 11126 84741 12% /boot

tmpfs tmpfs 371332 0 371332 0% /dev/shm

none binfmt\_misc 0 0 0 - /proc/sys/fs/binfmt\_misc

sunrpc rpc\_pipefs 0 0 0 - /var/lib/nfs/rpc\_pipefs

**实例 4**

将 /etc 底下的可用的磁盘容量以易读的容量格式显示

[root@www ~]# df -h /etc

Filesystem Size Used Avail Use% Mounted on

/dev/hdc2 9.5G 3.7G 5.4G 41% /

**du**

inux 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，还是和df命令有一些区别的，这里介绍Linux du命令。

语法：

du [-ahskm] 文件或目录名称

选项与参数：

* -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；

**实例 1**

列出目前目录下的所有文件容量

[root@www ~]# du

8 ./test4 <==每个目录都会列出来

8 ./test2

....中间省略....

12 ./.gconfd <==包括隐藏文件的目录

220 . <==这个目录(.)所占用的总量

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。

**实例 2**

将文件的容量也列出来

[root@www ~]# du -a

12 ./install.log.syslog <==有文件的列表了

8 ./.bash\_logout

8 ./test4

8 ./test2

....中间省略....

12 ./.gconfd

220 .

**实例 3**

检查根目录底下每个目录所占用的容量

[root@www ~]# du -sm /\*

7 /bin

6 /boot

.....中间省略....

0 /proc

.....中间省略....

1 /tmp

3859 /usr <==系统初期最大就是他了啦！

77 /var

通配符 \* 来代表每个目录。

与 df 不一样的是，du 这个命令其实会直接到文件系统内去搜寻所有的文件数据。

**fdisk**

fdisk 是 Linux 的磁盘分区表操作工具。

语法：

fdisk [-l] 装置名称

选项与参数：

* -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。

**实例 1**

列出所有分区信息

[root@AY120919111755c246621 tmp]# fdisk -l

Disk /dev/xvda: 21.5 GB, 21474836480 bytes

255 heads, 63 sectors/track, 2610 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

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

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x00000000

Device Boot Start End Blocks Id System

/dev/xvda1 \* 1 2550 20480000 83 Linux

/dev/xvda2 2550 2611 490496 82 Linux swap / Solaris

Disk /dev/xvdb: 21.5 GB, 21474836480 bytes

255 heads, 63 sectors/track, 2610 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

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

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x56f40944

Device Boot Start End Blocks Id System

/dev/xvdb2 1 2610 20964793+ 83 Linux

**实例 2**

找出你系统中的根目录所在磁盘，并查阅该硬盘内的相关信息

[root@www ~]# df / <==注意：重点在找出磁盘文件名而已

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/hdc2 9920624 3823168 5585388 41% /

[root@www ~]# fdisk /dev/hdc <==仔细看，不要加上数字喔！

The number of cylinders for this disk is set to 5005.

There is nothing wrong with that, but this is larger than 1024,

and could in certain setups cause problems with:

1) software that runs at boot time (e.g., old versions of LILO)

2) booting and partitioning software from other OSs

(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): <==等待你的输入！

输入 m 后，就会看到底下这些命令介绍

Command (m for help): m <== 输入 m 后，就会看到底下这些命令介绍

Command action

a toggle a bootable flag

b edit bsd disklabel

c toggle the dos compatibility flag

d delete a partition <==删除一个partition

l list known partition types

m print this menu

n add a new partition <==新增一个partition

o create a new empty DOS partition table

p print the partition table <==在屏幕上显示分割表

q quit without saving changes <==不储存离开fdisk程序

s create a new empty Sun disklabel

t change a partition's system id

u change display/entry units

v verify the partition table

w write table to disk and exit <==将刚刚的动作写入分割表

x extra functionality (experts only)

离开 fdisk 时按下 q，那么所有的动作都不会生效！相反的， 按下w就是动作生效的意思。

Command (m for help): p <== 这里可以输出目前磁盘的状态

Disk /dev/hdc: 41.1 GB, 41174138880 bytes <==这个磁盘的文件名与容量

255 heads, 63 sectors/track, 5005 cylinders <==磁头、扇区与磁柱大小

Units = cylinders of 16065 \* 512 = 8225280 bytes <==每个磁柱的大小

Device Boot Start End Blocks Id System

/dev/hdc1 \* 1 13 104391 83 Linux

/dev/hdc2 14 1288 10241437+ 83 Linux

/dev/hdc3 1289 1925 5116702+ 83 Linux

/dev/hdc4 1926 5005 24740100 5 Extended

/dev/hdc5 1926 2052 1020096 82 Linux swap / Solaris

# 装置文件名 启动区否 开始磁柱 结束磁柱 1K大小容量 磁盘分区槽内的系统

Command (m for help): q

想要不储存离开吗？按下 q 就对了！不要随便按 w 啊！

使用 p 可以列出目前这颗磁盘的分割表信息，这个信息的上半部在显示整体磁盘的状态。

**磁盘格式化**

磁盘分割完毕后自然就是要进行文件系统的格式化，格式化的命令非常的简单，使用 mkfs（make filesystem） 命令。

语法：

mkfs [-t 文件系统格式] 装置文件名

选项与参数：

* -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)

**实例 1**

查看 mkfs 支持的文件格式

[root@www ~]# mkfs[tab][tab]

mkfs mkfs.cramfs mkfs.ext2 mkfs.ext3 mkfs.msdos mkfs.vfat

按下两个[tab]，会发现 mkfs 支持的文件格式如上所示。

**实例 2**

将分区 /dev/hdc6（可指定你自己的分区） 格式化为 ext3 文件系统：

[root@www ~]# mkfs -t ext3 /dev/hdc6

mke2fs 1.39 (29-May-2006)

Filesystem label= <==这里指的是分割槽的名称(label)

OS type: Linux

Block size=4096 (log=2) <==block 的大小配置为 4K

Fragment size=4096 (log=2)

251392 inodes, 502023 blocks <==由此配置决定的inode/block数量

25101 blocks (5.00%) reserved for the super user

First data block=0

Maximum filesystem blocks=515899392

16 block groups

32768 blocks per group, 32768 fragments per group

15712 inodes per group

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912

Writing inode tables: done

Creating journal (8192 blocks): done <==有日志记录

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

# 这样就创建起来我们所需要的 Ext3 文件系统了！简单明了！

**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。

语法：

fsck [-t 文件系统] [-ACay] 装置名称

选项与参数：

* -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。

**实例 1**

查看系统有多少文件系统支持的 fsck 命令：

[root@www ~]# fsck[tab][tab]

fsck fsck.cramfs fsck.ext2 fsck.ext3 fsck.msdos fsck.vfat

**实例 2**

强制检测 /dev/hdc6 分区:

[root@www ~]# fsck -C -f -t ext3 /dev/hdc6

fsck 1.39 (29-May-2006)

e2fsck 1.39 (29-May-2006)

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

vbird\_logical: 11/251968 files (9.1% non-contiguous), 36926/1004046 blocks

如果没有加上 -f 的选项，则由于这个文件系统不曾出现问题，检查的经过非常快速！若加上 -f 强制检查，才会一项一项的显示过程。

**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。

磁盘挂载语法：

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

**实例 1**

用默认的方式，将刚刚创建的 /dev/hdc6 挂载到 /mnt/hdc6 上面！

[root@www ~]# mkdir /mnt/hdc6

[root@www ~]# mount /dev/hdc6 /mnt/hdc6

[root@www ~]# df

Filesystem 1K-blocks Used Available Use% Mounted on

.....中间省略.....

/dev/hdc6 1976312 42072 1833836 3% /mnt/hdc6

磁盘卸载命令 umount 语法：

umount [-fn] 装置文件名或挂载点

选项与参数：

* -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。

卸载/dev/hdc6

[root@www ~]# umount /dev/hdc6

# 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。

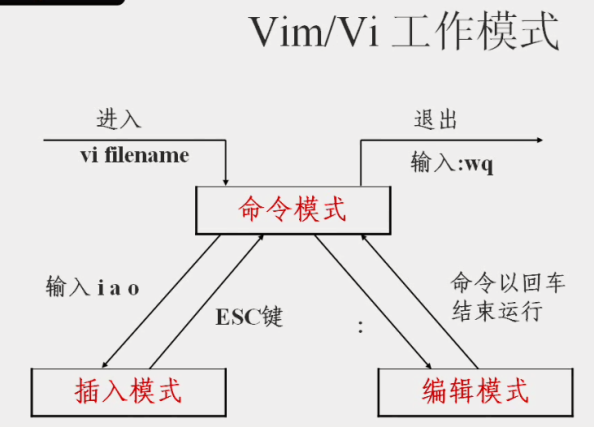
简单的来说， vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。 vim 则可以说是程序开发者的一项很好用的工具。 连 vim 的官方网站 (http://www.vim.org) 自己也说 vim 是一个程序开发工具而不是文字处理软件。

## vi/vim 的使用

基本上 vi/vim 共分为三种模式，分别是一般模式、编辑模式与指令列命令模式。 这三种模式的作用分别是：

* 一般模式：  
  以 vi 打开一个档案就直接进入一般模式了(这是默认的模式)。在这个模式中， 你可以使用『上下左右』按键来移动光标，你可以使用『删除字符』或『删除整行』来处理档案内容， 也可以使用『复制、贴上』来处理你的文件数据。
* 编辑模式：  
  在一般模式中可以进行删除、复制、贴上等等的动作，但是却无法编辑文件内容的！ 要等到你按下『i, I, o, O, a, A, r, R』等任何一个字母之后才会进入编辑模式。注意了！通常在 Linux 中，按下这些按键时，在画面的左下方会出现『INSERT 或 REPLACE 』的字样，此时才可以进行编辑。而如果要回到一般模式时， 则必须要按下『Esc』这个按键即可退出编辑模式。
* 指令列命令模式：  
  在一般模式当中，输入『 : / ? 』三个中的任何一个按钮，就可以将光标移动到最底下那一行。在这个模式当中， 可以提供你『搜寻资料』的动作，而读取、存盘、大量取代字符、离开 vi 、显示行号等等的动作则是在此模式中达成的！

简单的说，我们可以将这三个模式想成底下的图标来表示：



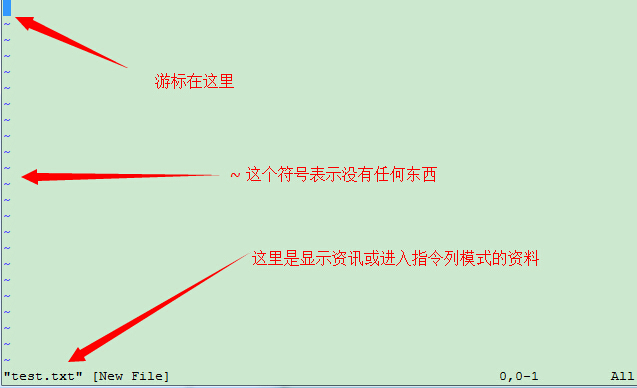
### vi/vim 使用实例

#### 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 test.txt 的文件时，你可以这样做：

[root@www ~]# vi test.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 进入编辑模式，开始编辑文字

在一般模式之中，只要按下 i, o, a 等字符就可以进入编辑模式了！

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。

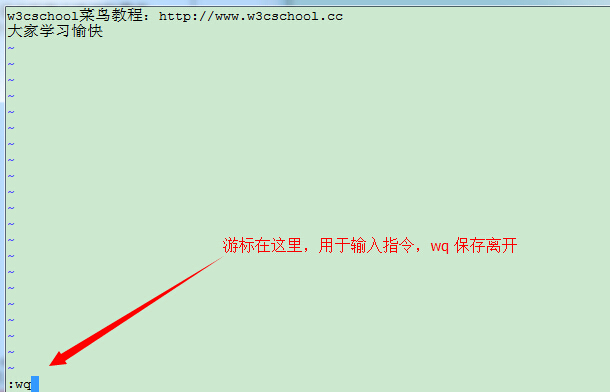


#### 按下 [ESC] 按钮回到一般模式

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 test.txt 的文件。是不是很简单。

## 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。

#### 第一部份：一般模式可用的按钮说明，光标移动、复制贴上、搜寻取代等

|  |  |
| --- | --- |
| **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列 |
| - | 光标移动到非空格符的上一列 |
| 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) |
| **搜寻与取代** | |
| /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 | 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,$s/word1/word2/g | 从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！(常用) |
| :1,$s/word1/word2/gc | 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列(常用) |
| ndd | n 为数字。删除光标所在的向下 n 列，例如 20dd 则是删除 20 列 (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 | 复制游标所在的那一行(常用) |
| nyy | n 为数字。复制光标所在的向下 n 列，例如 20yy 则是复制 20 列(常用) |
| y1G | 复制游标所在列到第一列的所有数据 |
| yG | 复制游标所在列到最后一列的所有数据 |
| 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 | 将光标所在列与下一列的数据结合成同一列 |
| 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) |
| 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) |

#### 第二部份：一般模式切换到编辑模式的可用的按钮说明

|  |  |
| --- | --- |
| **进入插入或取代的编辑模式** | |
| i, I | 进入插入模式(Insert mode)： i 为『从目前光标所在处插入』， I 为『在目前所在行的第一个非空格符处开始插入』。 (常用) |
| a, A | 进入插入模式(Insert mode)： a 为『从目前光标所在的下一个字符处开始插入』， A 为『从光标所在行的最后一个字符处开始插入』。(常用) |
| o, O | 进入插入模式(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插入新的一行』； O 为在目前光标所在处的上一行插入新的一行！(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) |

#### 第三部份：一般模式切换到指令列模式的可用的按钮说明

|  |  |
| --- | --- |
| **指令列的储存、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！若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开！ |
| :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 到指令列模式下执行 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』 对吧！ 数字加在动作之前，如我要向下移动 20 行呢？那就是『20j』或者是『20↓』即可。

**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。

基於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。

yum提供了查找、安装、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。

**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**操作的对象。

**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.列出所有可安裝的软件清单命令：yum list
* 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

**实例 1**

安装 pam-devel

[root@www ~]# yum install pam-devel

Setting up Install Process

Parsing package install arguments

Resolving Dependencies <==先检查软件的属性相依问题

--> Running transaction check

---> Package pam-devel.i386 0:0.99.6.2-4.el5 set to be updated

--> Processing Dependency: pam = 0.99.6.2-4.el5 for package: pam-devel

--> Running transaction check

---> Package pam.i386 0:0.99.6.2-4.el5 set to be updated

filelists.xml.gz 100% |=========================| 1.6 MB 00:05

filelists.xml.gz 100% |=========================| 138 kB 00:00

-> Finished Dependency Resolution

……(省略)

**实例 2**

移除 pam-devel

[root@www ~]# yum remove pam-devel

Setting up Remove Process

Resolving Dependencies <==同样的，先解决属性相依的问题

--> Running transaction check

---> Package pam-devel.i386 0:0.99.6.2-4.el5 set to be erased

--> Finished Dependency Resolution

Dependencies Resolved

=============================================================================

Package Arch Version Repository Size

=============================================================================

Removing:

pam-devel i386 0.99.6.2-4.el5 installed 495 k

Transaction Summary

=============================================================================

Install 0 Package(s)

Update 0 Package(s)

Remove 1 Package(s) <==还好，并没有属性相依的问题，单纯移除一个软件

Is this ok [y/N]: y

Downloading Packages:

Running rpm\_check\_debug

Running Transaction Test

Finished Transaction Test

Transaction Test Succeeded

Running Transaction

Erasing : pam-devel ######################### [1/1]

Removed: pam-devel.i386 0:0.99.6.2-4.el5

Complete!

**实例 3**

利用 yum 的功能，找出以 pam 为开头的软件名称有哪些？

[root@www ~]# yum list pam\*

Installed Packages

pam.i386 0.99.6.2-3.27.el5 installed

pam\_ccreds.i386 3-5 installed

pam\_krb5.i386 2.2.14-1 installed

pam\_passwdqc.i386 1.0.2-1.2.2 installed

pam\_pkcs11.i386 0.5.3-23 installed

pam\_smb.i386 1.1.7-7.2.1 installed

Available Packages <==底下则是『可升级』的或『未安装』的

pam.i386 0.99.6.2-4.el5 base

pam-devel.i386 0.99.6.2-4.el5 base

pam\_krb5.i386 2.2.14-10 base

**国内 yum 源**

网易（163）yum源是国内最好的yum源之一 ，无论是速度还是软件版本，都非常的不错。

将yum源设置为163 yum，可以提升软件包安装和更新的速度，同时避免一些常见软件版本无法找到。

**安装步骤**

首先备份/etc/yum.repos.d/CentOS-Base.repo

mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup

下载对应版本repo文件, 放入/etc/yum.repos.d/(操作前请做好相应备份)

* [CentOS5](http://mirrors.163.com/.help/CentOS5-Base-163.repo) ：http://mirrors.163.com/.help/CentOS5-Base-163.repo
* [CentOS6](http://mirrors.163.com/.help/CentOS6-Base-163.repo) ：http://mirrors.163.com/.help/CentOS6-Base-163.repo

运行以下命令生成缓存

yum clean all

yum makecache

除了网易之外，国内还有其他不错的yum源，比如中科大和搜狐。

中科大的yum源，安装方法查看：<https://lug.ustc.edu.cn/wiki/mirrors/help/centos>

sohu的yum源安装方法查看: <http://mirrors.sohu.com/help/centos.html>

# Shell 教程

Shell 是一个用C语言编写的程序，它是用户使用Linux的桥梁。Shell既是一种命令语言，又是一种程序设计语言。

Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核的服务。

Ken Thompson的sh是第一种Unix Shell，Windows Explorer是一个典型的图形界面Shell。

[**Shell 在线工具**](http://tool.w3cschool.cc/languages/online.php?language=bash)

## Shell 脚本

Shell 脚本（shell script），是一种为shell编写的脚本程序。

业界所说的shell通常都是指shell脚本，但读者朋友要知道，shell和shell script是两个不同的概念。

由于习惯的原因，简洁起见，本文出现的"shell编程"都是指shell脚本编程，不是指开发shell自身。

## Shell 环境

Shell 编程跟java、php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。

Linux的Shell种类众多，常见的有：

* Bourne Shell（/usr/bin/sh或/bin/sh）
* Bourne Again Shell（/bin/bash）
* C Shell（/usr/bin/csh）
* K Shell（/usr/bin/ksh）
* Shell for Root（/sbin/sh）
* ……

本教程关注的是 Bash，也就是 Bourne Again Shell，由于易用和免费，Bash在日常工作中被广泛使用。同时，Bash也是大多数Linux系统默认的Shell。

在一般情况下，人们并不区分 Bourne Shell 和 Bourne Again Shell，所以，像 **#!/bin/sh**，它同样也可以改为**#!/bin/bash**。

#!告诉系统其后路径所指定的程序即是解释此脚本文件的Shell程序。

## 第一个shell脚本

打开文本编辑器(可以使用vi/vim命令来创建文件)，新建一个文件test.sh，扩展名为sh（sh代表shell），扩展名并不影响脚本执行，见名知意就好，如果你用php写shell 脚本，扩展名就用php好了。

输入一些代码，第一行一般是这样：

#!/bin/bash

echo "Hello World !"

"#!" 是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种Shell。

echo命令用于向窗口输出文本。

### 运行Shell脚本有两种方法：

#### 1、作为可执行程序

将上面的代码保存为test.sh，并cd到相应目录：

chmod +x ./test.sh #使脚本具有执行权限

./test.sh #执行脚本

注意，一定要写成./test.sh，而不是test.sh，运行其它二进制的程序也一样，直接写test.sh，linux系统会去PATH里寻找有没有叫test.sh的，而只有/bin, /sbin, /usr/bin，/usr/sbin等在PATH里，你的当前目录通常不在PATH里，所以写成test.sh是会找不到命令的，要用./test.sh告诉系统说，就在当前目录找。

#### 2、作为解释器参数

这种运行方式是，直接运行解释器，其参数就是shell脚本的文件名，如：

/bin/sh test.sh

/bin/php test.php

这种方式运行的脚本，不需要在第一行指定解释器信息，写了也没用。

**Shell 变量**

定义变量时，变量名不加美元符号（$，PHP语言中变量需要），如：

your\_name="w3cschool.cc"

注意，变量名和等号之间不能有空格，这可能和你熟悉的所有编程语言都不一样。同时，变量名的命名须遵循如下规则：

* 首个字符必须为字母（a-z，A-Z）。
* 中间不能有空格，可以使用下划线（\_）。
* 不能使用标点符号。
* 不能使用bash里的关键字（可用help命令查看保留关键字）。

除了显式地直接赋值，还可以用语句给变量赋值，如：

for file in `ls /etc`

以上语句将 /etc 下目录的文件名循环出来。

**使用变量**

使用一个定义过的变量，只要在变量名前面加美元符号即可，如：

your\_name="qinjx"

echo $your\_name

echo ${your\_name}

变量名外面的花括号是可选的，加不加都行，加花括号是为了帮助解释器识别变量的边界，比如下面这种情况：

for skill in Ada Coffe Action Java do

echo "I am good at ${skill}Script"

done

如果不给skill变量加花括号，写成echo "I am good at $skillScript"，解释器就会把$skillScript当成一个变量（其值为空），代码执行结果就不是我们期望的样子了。

推荐给所有变量加上花括号，这是个好的编程习惯。

已定义的变量，可以被重新定义，如：

your\_name="tom"

echo $your\_name

your\_name="alibaba"

echo $your\_name

这样写是合法的，但注意，第二次赋值的时候不能写$your\_name="alibaba"，使用变量的时候才加美元符（$）。

**Shell 字符串**

字符串是shell编程中最常用最有用的数据类型（除了数字和字符串，也没啥其它类型好用了），字符串可以用单引号，也可以用双引号，也可以不用引号。单双引号的区别跟PHP类似。

**单引号**

str='this is a string'

单引号字符串的限制：

* 单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；
* 单引号字串中不能出现单引号（对单引号使用转义符后也不行）。

**双引号**

your\_name='qinjx'

str="Hello, I know your are \"$your\_name\"! \n"

双引号的优点：

* 双引号里可以有变量
* 双引号里可以出现转义字符

**拼接字符串**

your\_name="qinjx"

greeting="hello, "$your\_name" !"

greeting\_1="hello, ${your\_name} !"

echo $greeting $greeting\_1

**获取字符串长度**

string="abcd"

echo ${#string} #输出 4

**提取子字符串**

string="alibaba is a great company"

echo ${string:1:4} #输出liba

**查找子字符串**

string="alibaba is a great company"

echo `expr index "$string" is`

**注意：** 以上脚本中 "`" 是反引号，而不是单引号 "'"，不要看错了哦。

**Shell 数组**

bash支持一维数组（不支持多维数组），并且没有限定数组的大小。

类似与C语言，数组元素的下标由0开始编号。获取数组中的元素要利用下标，下标可以是整数或算术表达式，其值应大于或等于0。

**定义数组**

在Shell中，用括号来表示数组，数组元素用"空格"符号分割开。定义数组的一般形式为：

数组名=(值1 值2 ... 值n)

例如：

array\_name=(value0 value1 value2 value3)

或者

array\_name=(

value0

value1

value2

value3

)

还可以单独定义数组的各个分量：

array\_name[0]=value0

array\_name[1]=value1

array\_name[n]=valuen

可以不使用连续的下标，而且下标的范围没有限制。

**读取数组**

读取数组元素值的一般格式是：

${数组名[下标]}

例如：

valuen=${array\_name[n]}

使用@符号可以获取数组中的所有元素，例如：

echo ${array\_name[@]}

**获取数组的长度**

获取数组长度的方法与获取字符串长度的方法相同，例如：

# 取得数组元素的个数

length=${#array\_name[@]}

# 或者

length=${#array\_name[\*]}

# 取得数组单个元素的长度

lengthn=${#array\_name[n]}

**Shell 注释**

以"#"开头的行就是注释，会被解释器忽略。

sh里没有多行注释，只能每一行加一个#号。只能像这样：

#--------------------------------------------

# 这是一个自动打ipa的脚本，基于webfrogs的ipa-build书写：

# https://github.com/webfrogs/xcode\_shell/blob/master/ipa-build

# 功能：自动为etao ios app打包，产出物为14个渠道的ipa包

# 特色：全自动打包，不需要输入任何参数

#--------------------------------------------

##### 用户配置区 开始 #####

#

#

# 项目根目录，推荐将此脚本放在项目的根目录，这里就不用改了

# 应用名，确保和Xcode里Product下的target\_name.app名字一致

#

##### 用户配置区 结束 #####

如果在开发过程中，遇到大段的代码需要临时注释起来，过一会儿又取消注释，怎么办呢？

每一行加个#符号太费力了，可以把这一段要注释的代码用一对花括号括起来，定义成一个函数，没有地方调用这个函数，这块代码就不会执行，达到了和注释一样的效果。

## Shell echo命令

Shell 的 echo 指令与 PHP 的 echo 指令类似，都是用于字符串的输出。命令格式：

echo string

您可以使用echo实现更复杂的输出格式控制。

### 1.显示普通字符串:

echo "It is a test"

这里的双引号完全可以省略，以下命令与上面实例效果一致：

echo It is a test

### 2.显示转义字符

echo "\"It is a test\""

结果将是:

"It is a test"

同样，双引号也可以省略

### 3.显示变量

read 命令从标准输入中读取一行,并把输入行的每个字段的值指定给 shell 变量

#!/bin/sh

read name

echo "$name It is a test"

以上代码保存为 test.sh，name 接收标准输入的变量，结果将是:

[root@www ~]# sh test.sh

OK #标准输入

OK It is a test #输出

### 4.显示换行

echo -e "OK!\n" # -e 开启转义

echo "It it a test"

输出结果：

OK!

It it a test

### 5.显示不换行

#!/bin/sh

echo -e "OK! \c" # -e 开启转义 \c 不换行

echo "It is a test"

输出结果：

OK! It is a test

### 6.显示结果定向至文件

echo "It is a test" > myfile

### 7.原样输出字符串，不进行转义或取变量(用单引号)

echo '$name\"'

输出结果：

$name\"

### 8.显示命令执行结果

echo `date`

结果将显示当前日期

Thu Jul 24 10:08:46 CST 2014

# Shell test命令

Shell中的 test 命令用于检查某个条件是否成立，它可以进行数值、字符和文件三个方面的测试。

## 数值测试

|  |  |
| --- | --- |
| **参数** | **说明** |
| -eq | 等于则为真 |
| -ne | 不等于则为真 |
| -gt | 大于则为真 |
| -ge | 大于等于则为真 |
| -lt | 小于则为真 |
| -le | 小于等于则为真 |

实例演示：

num1=100

num2=100

if test $[num1] -eq $[num2]

then

echo 'The two numbers are equal!'

else

echo 'The two numbers are not equal!'

fi

输出结果：

The two numbers are equal!

## 字符串测试

|  |  |
| --- | --- |
| **参数** | **说明** |
| = | 等于则为真 |
| != | 不相等则为真 |
| -z 字符串 | 字符串长度伪则为真 |
| -n 字符串 | 字符串长度不伪则为真 |

实例演示：

num1=100

num2=100

if test num1=num2

then

echo 'The two strings are equal!'

else

echo 'The two strings are not equal!'

fi

输出结果：

The two strings are equal!

## 文件测试

|  |  |
| --- | --- |
| **参数** | **说明** |
| -e 文件名 | 如果文件存在则为真 |
| -r 文件名 | 如果文件存在且可读则为真 |
| -w 文件名 | 如果文件存在且可写则为真 |
| -x 文件名 | 如果文件存在且可执行则为真 |
| -s 文件名 | 如果文件存在且至少有一个字符则为真 |
| -d 文件名 | 如果文件存在且为目录则为真 |
| -f 文件名 | 如果文件存在且为普通文件则为真 |
| -c 文件名 | 如果文件存在且为字符型特殊文件则为真 |
| -b 文件名 | 如果文件存在且为块特殊文件则为真 |

实例演示：

cd /bin

if test -e ./bash

then

echo 'The file already exists!'

else

echo 'The file does not exists!'

fi

输出结果：

The file already exists!

另外，Shell还提供了与( -a )、或( -o )、非( ! )三个逻辑操作符用于将测试条件连接起来，其优先级为："!"最高，"-a"次之，"-o"最低。例如：

cd /bin

if test -e ./notFile -o ./bash

then

echo 'One file exists at least!'

else

echo 'Both dose not exists!'

fi

输出结果：

One file exists at least!

**Shell 流程控制**

和Java、PHP等语言不一样，sh的流程控制不可为空，如(以下为PHP流程控制写法)：

<?php

if (isset($\_GET["q"])) {

search(q);

}

else {

//do nothing

}

在sh/bash里可不能这么写，如果else分支没有语句执行，就不要写这个else，就像这样

**if else**

**if**

if 语句语法格式：

if condition

then

command1

command2

...

commandN

fi

写成一行（适用于终端命令提示符）：

if `ps -ef | grep ssh`; then echo hello; fi

末尾的fi就是if倒过来拼写，后面还会遇到类似的。

**if else**

if else 语法格式：

if condition

then

command1

command2

...

commandN

else

command

fi

**if else-if else**

if else-if else 语法格式：

if condition1

then

command1

elif condition2

command2

else

commandN

fi

if else语句经常与test命令结合使用，如下所示：

num1=$[2\*3]

num2=$[1+5]

if test $[num1] -eq $[num2]

then

echo 'The two numbers are equal!'

else

echo 'The two numbers are not equal!'

fi

输出结果：

The two numbers are equal!

**for 循环**

与其他编程语言类似，Shell支持for循环。

for循环一般格式为：

for var in item1 item2 ... itemN

do

command1

command2

...

commandN

done

写成一行：

for var in item1 item2 ... itemN; do command1; command2… done;

当变量值在列表里，for循环即执行一次所有命令，使用变量名获取列表中的当前取值。命令可为任何有效的shell命令和语句。in列表可以包含替换、字符串和文件名。

in列表是可选的，如果不用它，for循环使用命令行的位置参数。

例如，顺序输出当前列表中的数字：

for loop in 1 2 3 4 5

do

echo "The value is: $loop"

done

输出结果：

The value is: 1

The value is: 2

The value is: 3

The value is: 4

The value is: 5

顺序输出字符串中的字符：

for str in 'This is a string'

do

echo $str

done

输出结果：

This is a string

**while 语句**

while循环用于不断执行一系列命令，也用于从输入文件中读取数据；命令通常为测试条件。其格式为：

while condition

do

command

done

命令执行完毕，控制返回循环顶部，从头开始直至测试条件为假。

以下是一个基本的while循环，测试条件是：如果COUNTER小于5，那么条件返回真。COUNTER从0开始，每次循环处理时，COUNTER加1。运行上述脚本，返回数字1到5，然后终止。

COUNTER=0

while [ $COUNTER -lt 5 ]

do

COUNTER='expr $COUNTER+1'

echo $COUNTER

done

运行脚本，输出：

1

2

3

4

5

while循环可用于读取键盘信息。下面的例子中，输入信息被设置为变量FILM，按<Ctrl-D>结束循环。

echo 'type <CTRL-D> to terminate'

echo -n 'enter your most liked film: ''

while read FILM

do

echo "Yeah! great film the $FILM"

done

运行脚本，输出类似下面：

type <CTRL-D> to terminate

enter your most liked film: Sound of Music

Yeah! great film the Sound of Music

**无限循环**

无限循环语法格式：

while :

do

command

done

或者

while true

do

command

done

或者

for (( ; ; ))

**until 循环**

until循环执行一系列命令直至条件为真时停止。

until循环与while循环在处理方式上刚好相反。

一般while循环优于until循环，但在某些时候—也只是极少数情况下，until循环更加有用。

until 语法格式:

until condition

do

command

done

条件可为任意测试条件，测试发生在循环末尾，因此循环至少执行一次—请注意这一点。

**case**

Shell case语句为多选择语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

case 值 in

模式1)

command1

command2

...

commandN

;;

模式2）

command1

command2

...

commandN

;;

esac

case工作方式如上所示。取值后面必须为单词in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 \* 捕获该值，再执行后面的命令。

下面的脚本提示输入1到4，与每一种模式进行匹配：

echo 'Input a number between 1 to 4'

echo 'Your number is:\c'

read aNum

case $aNum in

1) echo 'You select 1'

;;

2) echo 'You select 2'

;;

3) echo 'You select 3'

;;

4) echo 'You select 4'

;;

\*) echo 'You do not select a number between 1 to 4'

;;

esac

输入不同的内容，会有不同的结果，例如：

Input a number between 1 to 4

Your number is:3

You select 3

**跳出循环**

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，Shell使用两个命令来实现该功能：break和continue。

**break命令**

break命令允许跳出所有循环（终止执行后面的所有循环）。

下面的例子中，脚本进入死循环直至用户输入数字大于5。要跳出这个循环，返回到shell提示符下，需要使用break命令。

#!/bin/bash

while :

do

echo -n "Input a number between 1 to 5: "

read aNum

case $aNum in

1|2|3|4|5) echo "Your number is $aNum!"

;;

\*) echo "You do not select a number between 1 to 5, game is over!"

break

;;

esac

done

**continue**

continue命令与break命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。

对上面的例子进行修改：

#!/bin/bash

while :

do

echo -n "Input a number between 1 to 5: "

read aNum

case $aNum in

1|2|3|4|5) echo "Your number is $aNum!"

;;

\*) echo "You do not select a number between 1 to 5!"

continue

echo "Game is over!"

;;

esac

done

运行代码发现，当输入大于5的数字时，该例中的循环不会结束，语句 **echo "Game is over!"** 永远不会被执行。

**esac**

case的语法和C family语言差别很大，它需要一个esac（就是case反过来）作为结束标记，每个case分支用右圆括号，用两个分号表示break。

# Shell 函数

linux shell 可以用户定义函数，然后在shell脚本中可以随便调用。

shell中函数的定义格式如下：

[ function ] funname [()]

{

action;

[return int;]

}

说明：

* 1、可以带function fun() 定义，也可以直接fun() 定义,不带任何参数。
* 2、参数返回，可以显示加：return 返回，如果不加，将以最后一条命令运行结果，作为返回值。 return后跟数值n(0-255

下面的例子定义了一个函数并进行调用：

#!/bin/bash

demoFun(){

echo "This is your first shell function!"

}

echo "Function begin..."

demoFun

echo "Function end!

输出结果：

Function begin...

This is your first shell function!

Function end!

下面定义一个带有return语句的函数：

#!/bin/bash

funWithReturn(){

echo "The function is to get the sum of two numbers..."

echo -n "Input first number: "

read aNum

echo -n "Input another number: "

read anotherNum

echo "The two numbers are $aNum and $anotherNum !"

return $(($aNum+$anotherNum))

}

funWithReturn

echo "The sum of two numbers is $? !"

输出类似下面：

The function is to get the sum of two numbers...

Input first number: 25

Input another number: 50

The two numbers are 25 and 50 !

The sum of two numbers is 75 !

函数返回值在调用该函数后通过 $? 来获得。

注意：所有函数在使用前必须定义。这意味着必须将函数放在脚本开始部分，直至shell解释器首次发现它时，才可以使用。调用函数仅使用其函数名即可。

## 函数参数

在Shell中，调用函数时可以向其传递参数。在函数体内部，通过 $n 的形式来获取参数的值，例如，$1表示第一个参数，$2表示第二个参数...

带参数的函数示例：

#!/bin/bash

funWithParam(){

echo "The value of the first parameter is $1 !"

echo "The value of the second parameter is $2 !"

echo "The value of the tenth parameter is $10 !"

echo "The value of the tenth parameter is ${10} !"

echo "The value of the eleventh parameter is ${11} !"

echo "The amount of the parameters is $# !"

echo "The string of the parameters is $\* !"

}

funWithParam 1 2 3 4 5 6 7 8 9 34 73

输出结果：

The value of the first parameter is 1 !

The value of the second parameter is 2 !

The value of the tenth parameter is 10 !

The value of the tenth parameter is 34 !

The value of the eleventh parameter is 73 !

The amount of the parameters is 12 !

The string of the parameters is 1 2 3 4 5 6 7 8 9 34 73 !"

注意，$10 不能获取第十个参数，获取第十个参数需要${10}。当n>=10时，需要使用${n}来获取参数。

另外，还有几个特殊字符用来处理参数：

|  |  |
| --- | --- |
| **参数处理** | **说明** |
| $# | 传递到脚本的参数个数 |
| $\* | 以一个单字符串显示所有向脚本传递的参数 |
| $$ | 脚本运行的当前进程ID号 |
| $! | 后台运行的最后一个进程的ID号 |
| $@ | 与$#相同，但是使用时加引号，并在引号中返回每个参数。 |
| $- | 显示Shell使用的当前选项，与set命令功能相同。 |
| $? | 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。 |